NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DESIGN AND PERFORMANCE ANALYSIS OF AN ASYNCHRONOUS PIPELINED MULTIPLIER WITH COMPARISON TO SYNCHRONOUS IMPLEMENTATION

by

Kirk A. Shawhan

December 2000

Thesis Advisor: Douglas J. Fouts
Co-Advisor: Herschel H. Loomis, Jr.

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 2000	Master's Thesis

4. TITLE AND SUBTITLE :
Design and Performance Analysis of an Asynchronous Pipelined Multiplier with Comparison to Synchronous Implementation

5. FUNDING NUMBERS

6. AUTHOR(S)
Shawhan, Kirk A.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING /MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Synchronous techniques have dominated digital logic system design for decades because they are well understood and less complicated to implement. With the advent of more exotic high-speed transistors, the issues of clock skew, system performance, power consumption, and technology migration become critical to synchronous system designers. Asynchronous digital design techniques utilize a local completion signal or request/acknowledge handshake to lend the stability afforded by the global clock in synchronous systems. This research evaluates a moderately complex digital system, an 8x8-bit multiplier utilizing high-speed Indium Phosphide heterostructure bipolar junction transistors, to determine whether asynchronous logic design can compete with synchronous design in terms of system speed and power consumption. Theoretical timing equations are developed that relate the relative merits of each technique for input-to-output latency and system throughput. Tanner SPICE simulation tools are used to evaluate the full 8x8-bit asynchronous array multiplier. Finally, direct comparisons are made between five separate pipelined configurations of the multiplier utilizing both synchronous and asynchronous timing methodologies. As integrated circuits become smaller, faster, and more complex, asynchronous schemes will continue to mature and become more prevalent in digital system design.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Asynchronous Logic, Pipelined Multiplier, Micropipelines	94
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

DESIGN AND PERFORMANCE ANALYSIS OF AN ASYNCHRONOUS PIPELINED MULTIPLIER WITH COMPARISON TO SYNCHRONOUS IMPLEMENTATION

Kirk A. Shawhan
Lieutenant Colonel, United States Marine Corps
B.S., University of Notre Dame, 1984

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2000

Author: Kirk A. Shawhan
Kirk A. Shawhan

Approved by: Douglas J. Fouts
Douglas J. Fouts, Thesis Advisor

Herschel H. Loomis, Jr., Co-Advisor

Jeffrey B. Knorr, Chairman
Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Synchronous techniques have dominated digital logic system design for decades because they are well understood and less complicated to implement. With the advent of more exotic high-speed transistors, the issues of clock skew, system performance, power consumption, and technology migration become critical to synchronous system designers. Asynchronous digital design techniques utilize a local completion signal or request/acknowledge handshake to lend the stability afforded by the global clock in synchronous systems. This research evaluates a moderately complex digital system, an 8x8-bit multiplier utilizing high-speed Indium Phosphide heterostructure bipolar junction transistors, to determine whether asynchronous logic design can compete with synchronous design in terms of system speed and power consumption. Theoretical timing equations are developed that relate the relative merits of each technique for input-to-output latency and system throughput. Tanner SPICE simulation tools are used to evaluate the full 8x8-bit asynchronous array multiplier. Finally, direct comparisons are made between five separate pipelined configurations of the multiplier utilizing both synchronous and asynchronous timing methodologies. As integrated circuits become smaller, faster, and more complex, asynchronous schemes will continue to mature and become more prevalent in digital system design.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Synchronous logic design techniques have dominated the arena of digital design for over thirty years. As the negative impacts of clock skew, worst-case system performance, power consumption, system modularity, and technology migration push efficient synchronous design to the limits, system engineers have looked to other techniques for digital implementation. Asynchronous logic design concepts have been studied for decades and are now beginning to show promise for both commercial and military applications.

A moderately complex digital system, in this case an 8x8-bit pipelined multiplier, was designed using the asynchronous micropipeline technique with a four-cycle request/acknowledge handshake. High-speed Indium Phosphide heterostructure bipolar junction transistors form the basis for the digital logic gates in the design. In theory, latency performance for asynchronous systems compares favorably to synchronous implementation; however, the overhead involved in the four-cycle handshake limits throughput performance. Theoretically, asynchronous logic displays improved overall power consumption performance compared to synchronous logic.

For his Naval Postgraduate School Electrical Engineering Masters Thesis, Major John Calvert[1], United States Marine Corps, has designed and tested a similar pipelined multiplier using optimum synchronous techniques. Tanner SPICE simulations of the asynchronous and synchronous models follow the theoretical trends described. Latency

[1] Calvert, J.R., *Design of a Synchronous Pipelined Multiplier and Analysis of Clock Skew in High-Speed Digital Systems*, Masters Thesis, Naval Postgraduate School, Monterey, CA, Dec 2000.

performance of the two systems is relatively similar through all five pipelined designs: one, two, four, six and ten stage implementations. For the one-stage system, the synchronous system is 40% faster than the asynchronous design in terms of throughput performance. For the ten-stage multiplier, the synchronous implementation is 1200% faster, 5.56×10^9 multiplies-per-second versus 2.50×10^9 multiplies-per-second. Power consumption of the one-stage asynchronous system is 30% lower than the synchronous version while the ten-stage asynchronous multiplier consumes 34% less power. The overall device count of the synchronous system as designed is considerably lower than the asynchronous multiplier due primarily to the types of memory elements chosen for implementation. Major Calvert has chosen a current mode latch as the basis of the D Flip-Flop used in the synchronous design. The asynchronous design utilizes cross-coupled NOR gate latches in the stage registers which require ten more devices per element to implement. Therefore, the asynchronous design will require a larger, more complex layout.

An in-depth study of the merits of asynchronous versus synchronous design techniques has been accomplished through this research. Although not as technologically mature as synchronous logic design, asynchronous techniques display potential for the future, especially when system power consumption is a primary concern.

ACKNOWLEDGMENT

The time and effort represented by this thesis are a testament to the patience, support and loyalty of my wife, Mary. This research would not have been completed without her positive outlook and backing and the support of my children, Thomas, Mary Catherine and Daniel.

I would also like to thank and acknowledge my thesis advisors, Professor Fouts and Professor Loomis. Quick to jump into the foxhole to work through a design problem or timing equation, they are true academic role models and experts in the field of digital logic design.

Finally, I cannot extend enough thanks and gratitude to my numerous class and research partners who helped make the Naval Postgraduate School such a positive educational experience. More specifically, I would like to thank Major John Calvert of the United States Marine Corps and Lieutenant Karl Eimers of the United States Navy for keeping me focused on the academic tasks at hand.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PURPOSE OF RESEARCH

The purpose of this research is to determine whether there is a better method for building a high clockrate digital system other than the clocked, synchronous method primarily used in today's Complementary Metal Oxide Semiconductor (CMOS) digital systems. As logic propagation delays become smaller, and especially for systems implemented with exotic, high-speed technology such as Indium Phosphide (InP), companies that design and fabricate digital systems are running into physical and timing limitations. Circuit designers are continuously looking for methods to build a better mousetrap. One technology that has shown promise is asynchronous or self-timed logic. The goal of this thesis will be to utilize a moderately complex circuit, an 8x8-bit multiplier, to compare the speed, size, and power consumption of synchronous versus asynchronous high-speed logic.

B. DEVICE SPEED

Digital design is driven by many factors not the least of which is device speed. Whether the processor is a personal computer, a Digital Signal Processor (DSP), or an embedded controller, one of the first specifications a

customer or system engineer will consider is the frequency at which the device will operate. The speed of the system is determined by the minimum length of time at which the components in the systems can operate and continue to produce valid results. To increase speed, the design engineer must at a minimum consider the physical capabilities of the underlying transistor technology and the length of the critical logic path in the system.

Indium Phosphide heterostructure bipolar junction transistors (InP HBTs) produced by the Hughes Research Laboratory are utilized as the basis for this research. Logic gates designed with InP HBTs have a much shorter propagation delay than logic gates developed with the CMOS transistors used most often in digital logic design. Designs incorporating InP HBTs are, therefore, capable of higher operating frequencies.

Single logic gates do not determine the overall speed of an IC. Designers make use of gates in series and parallel to perform the desired function. The critical path is typically the longest series of logic gates which is unbroken by memory elements or registers. The minimum operating period of the device will be driven by the total propagation delay from the start to the end of this path. Historically, design engineers have increased the speed of processors by pipelining these paths. Pipelining involves splitting up the length of the critical path by

systematically introducing memory elements (Figure 1). When pipelining a circuit, consideration must be given to design trade-offs that include the increased layout area and power consumption required by the additional memory elements.

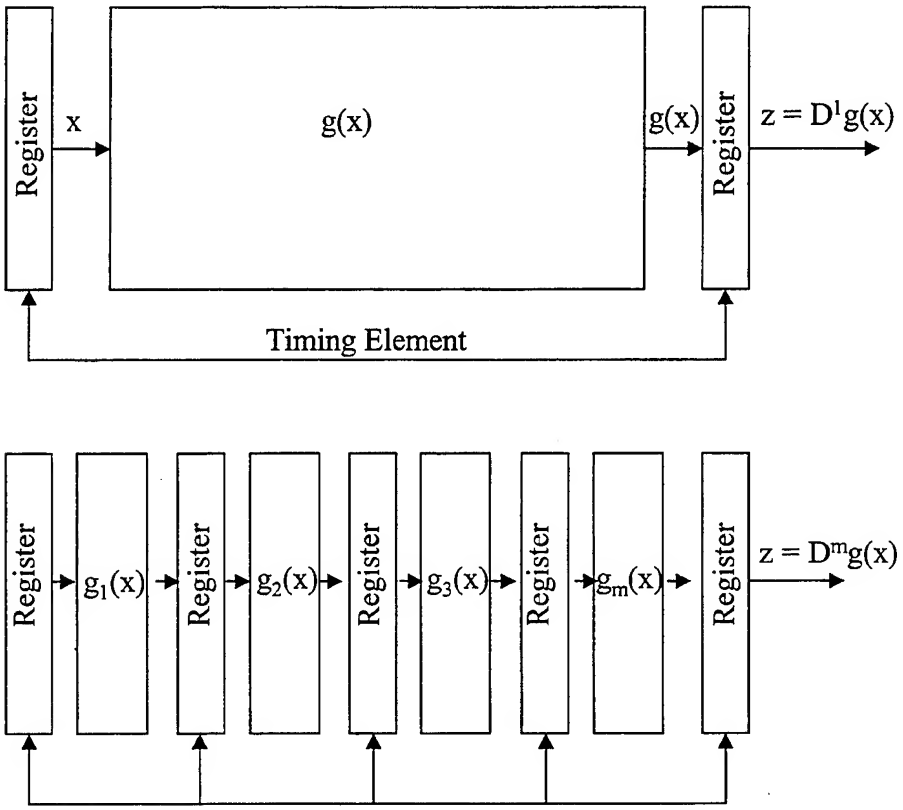

Figure 1. Logic Pipelining

(From Ref.[1])

C. PIPELINED MULTIPLIER

Digital Signal Processors are embedded in numerous technologies in today's commercial and defense markets. For example, overhead collection platforms require extremely capable high speed DSPs to provide data

compression/decompression and data encryption/decryption to produce and securely transmit the high volume of real-time traffic required for today's technologically oriented battlefield. The arithmetic logic units (ALU) and digital multipliers performing these calculations are at the heart of any DSP. In a future network-centric battle, victory may be determined by which side can move data most efficiently and accurately; who can add, subtract, multiply and divide the quickest.

An 8x8-bit multiplier is a moderately complex digital circuit that lends itself well to the purpose of this research. It can be sufficiently pipelined in a number of configurations to make comparisons both within a logic class (asynchronous vs. asynchronous) and across logic classes (synchronous vs. asynchronous). The 8x8-bit multiplier can also be quickly scaled to produce the 16x16-bit or larger configurations found in many DSPs.

D. RELATED WORK

There is an extensive body of work available for review in synchronous design techniques. Computer aided design (CAD) tools contain large libraries of synchronous components and cells which circuit designers routinely use (Ref.[2]). Major John Calvert of the Naval PostGraduate School has designed a synchronous implementation of the 8x8-bit pipelined multiplier that is utilized for direct

comparison to the asynchronous design of this research (Ref.[3]).

There is also a large body of work available for reference concerning the potential advantages of asynchronous design (Ref.[2]). Asynchronous circuits have been designed and tested, primarily in CMOS technology, for comparison to synchronous realizations (Refs. [4], [5], [6], and [7]). However, no work could be found which implements the type of high-speed technology represented by InP HBTs. The increased frequency capability of these transistors highlights some of the disadvantages of synchronous design such as clock skew. This research clearly demonstrates the potential for asynchronous realizations in future systems.

E. THESIS ORGANIZATION

The second chapter of this thesis concentrates on the concepts of synchronous and asynchronous digital design. The advantages and disadvantages of the two technologies are addressed. Several asynchronous design methodologies are also described.

Chapter III breaks down elements of asynchronous or self-timed pipelined design. The handshake process and control circuitry required are detailed in this portion of the thesis.

Multiplier design and pipeline considerations are covered in Chapter IV. Chapter IV also includes the

simulation results for each implementation of the self-timed pipelined multiplier using InP HBTs. Tanner SPICE (TSPICE) schematic simulation model results focus on the latency, throughput, power consumption and overall device counts realized for each implementation.

In Chapter V, synchronous design results are compared directly to these asynchronous outcomes. At this point, the primary question of this research is revealed; is it possible to build a better system?

II. BACKGROUND

A. SYNCHRONOUS DESIGN

1. Methodology

Synchronous logic designers rely on a clock signal to lend stability and discipline to their systems. The clock controls data flow and result storage and allows the system engineer to design with little regard to hazards and feedback (Ref. [8]).

Figure 2 depicts a process that incorporates synchronous design techniques. Operands are fed into a system of logic gates and clocked into a storage element, in this case a register, after the end result has stabilized. This result will not be output until all logic transients have settled. The clock period is given by the following:

$$T_{clock} \geq T_P + T_{REG} + T_{skew} \qquad \textbf{Equation 1}$$

where T_P is determined by the worst-case propagation delay of the process and T_{REG} is given by the propagation delay and set-up time of the register. Worst-case or maximum propagation delays are utilized so that the system will avoid any metastability that could occur in the process. The latency of this system is defined as the time it takes one set of operands to propagate to the output and is equal to T_{clock} in this example. The throughput of the system is the

rate at which the results will appear at the output, in this case $1/T_{clock}$.

Figure 3 represents a pipelined implementation of the

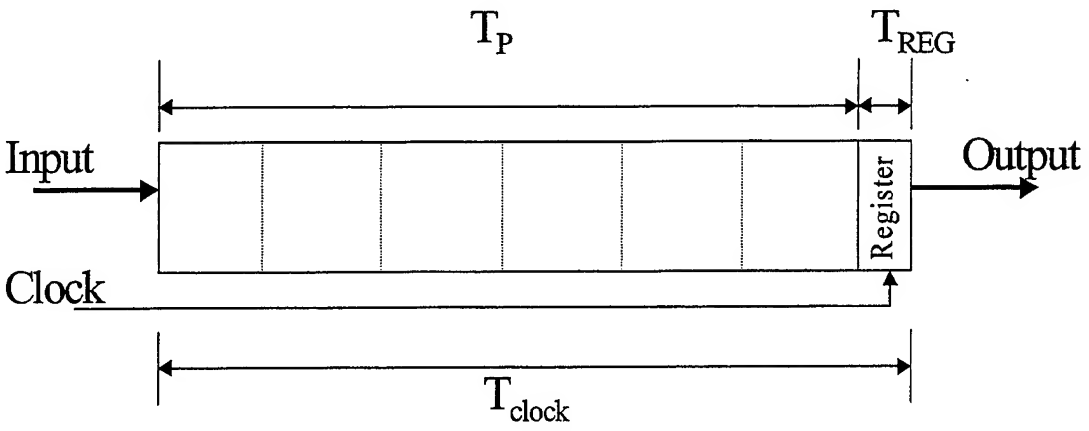

Figure 2. Non-pipelined Synchronous Process

(After Ref.[9])

process depicted in Figure 2. The clock period is determined by:

$$T_{clock} \geq T_S + T_{REG} + T_{skew} \qquad \textbf{Equation 2}$$

where T_s is the worst-case section propagation delay and T_{skew} is the clock skew which will be discussed in detail. The latency of this system is determined by the degree to which the original process is pipelined. If there are k slices, then latency is $k*T_{clock}$, which will be longer than the original latency due to the added register delays. Throughput of a full pipeline is still determined by $1/T_{clock}$, which is now a higher frequency. It is important to note

that the sections are not required to be of equal propagation length. However, the clock period is determined

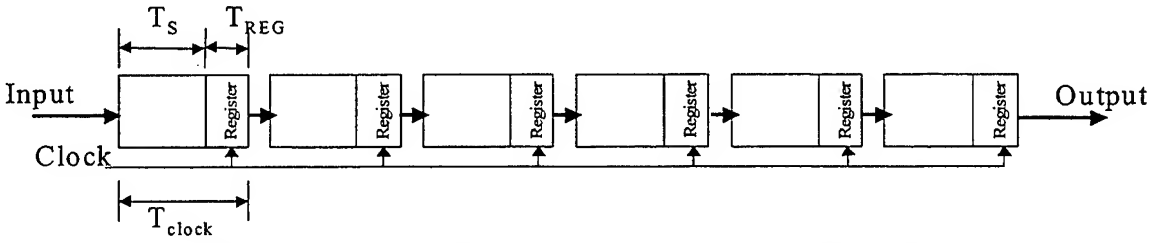

Figure 3. Pipelined Synchronous Process

(After Ref.[9])

by the section with the longest propagation delay; therefore, it is most efficient to slice the process into similar length delay paths.

2. Advantages

Synchronous techniques have been the backbone of design for decades because they work. System engineers in academic and commercial environments have studied these methods in detail and synchronous technology is generally understood. As previously stated, the bulk of computer-aided design (CAD) systems have large component libraries of synchronous circuits and cells ready for use. Synchronous design techniques have matured along with device fabrication technologies to produce reliable high clockrate systems. (Ref.[2])

3. Disadvantages

The small size and high complexity of today's IC systems highlight the disadvantages of synchronous design

techniques. These include: clock skew, worst-case system performance, power consumption, modularity, and technology migration (Refs. [2], [8], and [10]).

a) Clock Skew

Synchronous technology requires a global clock system to provide the discipline and order that make complex digital systems work. On large system-on-a-chip ICs, the clock distribution tree takes up an inordinate amount of layout area and consumes a large percentage of power. Due to the inaccuracy of current fabrication techniques to precisely reproduce components and cells on a single wafer, the required global clock transitions will not occur at the same time throughout the system. The propagation delay of metal interconnects also effects clock timing across the IC and enhances the size of the deviation from precise synchronization. Clock skew is a measure of this deviation.

For slower systems, clock skew is largely ignored because it is a small fraction of section or register propagation delay and therefore can be accounted for with a small clock timing safety margin. However, as feature size is reduced and systems operate at higher frequencies, clock skew becomes a larger percentage of the theoretical clock period. Designers expend time and effort dealing with clock skew problems to fully optimize their synchronous systems. Clock skew is especially critical in two and four phase

clock systems that rely on stringent clock signal overlap constraints to work properly.

b) System Performance

The clock period of a synchronous system is based on the worst-case propagation delay of the critical path in the design. This worst-case propagation delay can be effected by a number of factors to include: the fabrication process, system temperature, and supply voltage. The system engineer must have reliable chip test facilities to accurately measure the worst case scenario. Typically, this scenario occurs only a fraction of the time; however, the engineer must utilize this situation to design the global clock. Throughput and latency optimization is impacted by this worst-case criteria.

c) Power Consumption

A good analogy for the power consumption of synchronous systems is the operation of traffic lights late at night. Whether there is traffic to service or not, stoplights will continue to switch from green to yellow to red, wasting energy. The same can be said of synchronous digital systems; the clock will continue to operate and the entire IC will continue to consume power whether there are operands in the pipeline or not. This is critical in CMOS

design because these clocked transitions drive peak power consumption.

d) Modularity

Synchronous designs are difficult to connect to the outside world that is asynchronous by nature. Asynchronous inputs to a synchronous design can lead to metastable states in the input registers or first stage of logic. There are methods to decrease the likelihood of metastability; however, none are guaranteed. Interconnecting synchronous systems that are running at different clock frequencies can also be a difficult engineering task.

e) Technology Migration

One method of increasing the speed of an IC is to decrease the size of the layout area. This decrease in size typically realizes shorter propagation delays for system components and higher yields during wafer fabrication. However, when scaling a process, the system designer must again determine the critical path and reengineer the global clock signal. This process is time consuming and financially costly.

Integrated circuits can also be improved in parts. A redesign of one portion of a system is possible to realize overall gains. In a synchronous system, partial redesign is not a worthwhile exercise unless the critical path of the

system is targeted. For instance, a system engineer would not redesign a DSP multiplier if the instruction decode section was the critical path of the processor.

B. ASYNCHRONOUS DESIGN

1. Methodology

Asynchronous circuit design has been a topic of research since the infancy of the IC technology field. However, the dominance of synchronous techniques has often overshadowed asynchronous methods. The disadvantages of synchronous designs highlighted previously call for a closer investigation of asynchronous technology.

a) Truly Asynchronous

Truly asynchronous systems are speed independent and delay insensitive; they produce outputs when the process is complete, not before or after. Asynchronous systems that are speed independent and delay insensitive are extremely difficult to design and implement. Digital logic components and circuits do have critical paths and differential timing considerations that must be taken into account when designing truly asynchronous systems. Asynchronous systems must be free of static and dynamic hazards and critical races. The size and complexity of today's ICs makes truly asynchronous design virtually impossible. Engineering academia has historically recommended synchronization of

asynchronous inputs followed by synchronous state machines (Refs.[7] and [9]).

b) Two-rail System

One asynchronous method that has shown promise is a two-rail implementation. The two-rail system codes logic outputs to allow for signaling of operation completion. The state table for a sample two-rail system is displayed in Table 1. In this example, operation completion is signaled by the appropriate 01 or 10. To function properly, it must return to the null state (00) before the next set of operands can be loaded for evaluation. (Ref.[11])

Item	Representation
Acknowledge-Null	00
Data Value-logical 0	01
Data Value-logical 1	10
Not Allowed	11

Table 1. Two-rail System State Table

c) Micropipeline

The micropipeline method represents a self-timed implementation of asynchronous design. This method relies on a locally timed handshake protocol to control operand propagation. Figure 4 represents the micropipeline

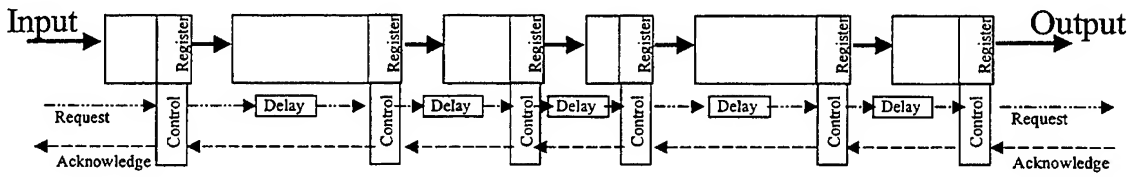

Figure 4. Asynchronous Micropipeline

(After Refs.[2] and [12])

technique. A request or data valid signal is transmitted to the register when the inputs to the register have stabilized after the required delay. The register cannot accept inputs until the following register acknowledges receipt of the previous data. This local handshake scheme ensures that only valid data will be forwarded through the pipeline (Refs.[8] and [12]).

The micropipeline handshake is usually either a two-cycle or four-cycle implementation (Figures 5, 6). The two-cycle protocol is a signal transition scheme that has benefits when considering CMOS applications because it does not have multiple line transitions for each handshake. Theoretically, this technique lends itself to speed and

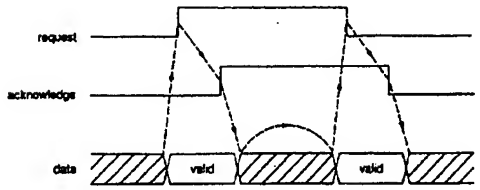

Figure 5. Two-cycle Handshake

(From Ref.[2])

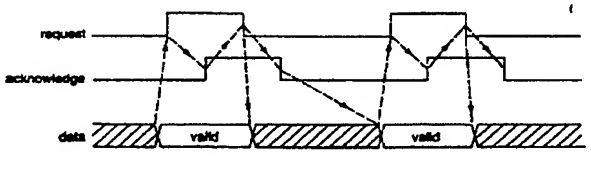

Figure 6. Four-cycle Handshake

(From Ref.[2])

power consumption benefits. However, the complex signal transition control of two-cycle handshaking requires event driven latches that are component intensive and difficult to design in the current-mode logic (CML) utilized in this research. The four-cycle protocol is simple to implement and requires fewer components than the two-cycle routine. (Refs.[2] and [13])

The delay element in the control path allows for valid operation of the micropipeline. This time delay in conjunction with the inherent control delay of the appropriate handshake scheme allows for the operands to be accurately manipulated before output data is clocked into the memory element. In this manner, the micropipeline scheme is very similar to synchronous techniques; worst-case timing delays are utilized to insure hazard avoidance. However, these worst-case scenarios are locally evaluated allowing for much greater system flexibility. As indicated in Figure 4, the propagation delays of individual sections of the pipeline are critical because the control delays are modeled after these local requirements. At a minimum, micropipelines display improved latency behavior when compared to synchronous implementations.

2. Advantages

a) Clock Skew

Asynchronous logic does not rely on a global clock signal to maintain system stability; therefore, clock skew is not a factor. Micropipelines have issues with the timing and propagation delay for the handshake control signals; however, these are local timing issues and they are simpler to handle on a small scale.

b) System Performance

Asynchronous logic typically displays average case propagation delay through the pipeline vice the worst-case scenario of synchronous logic. For truly asynchronous and two-rail systems when the logic gates in the pipeline have completed their tasks, the data is moved along. Micropipelines will typically utilize worst-case control delays; however, these are local pipeline path issues vice a system-wide worst-case concern.

c) Power Consumption

The stoplight analogy is valid for asynchronous logic. The lights will only operate when there is traffic to pass; otherwise, the system is idle. This is especially critical in CMOS technology because the idle state consumes very low power. To take the analogy one step further, asynchronous logic is more efficient because it allows the

traffic to continue through a set of properly timed lights. Traffic lights in a synchronous system will blink on and off in conjunction with the longest time in the entire system that it takes to cross between two intersections. Data crossing a short path must wait for the entire worst-case delay. In an asynchronous system, the lights will be timed for the actual propagation delay between each intersection.

d) Modularity

Asynchronous systems can be built in small stages and connected without regard to a global clock. As long as the appropriate design constraints have been met and the handshakes are in place, the stages will piece together nicely. Asynchronous techniques are also useful for interfaces with the asynchronous real world.

e) Technology Migration

Truly asynchronous systems are scalable without any regards to clock signals which saves in development time. For a micropipeline system, improvement in one path of the circuit can lead to overall quality gains in the IC. A shorter propagation delay in one stage will improve entire system latency; therefore, it is worthwhile to redesign any path in the system.

3. Disadvantages

The advantages of asynchronous design make it appear to have the future design market cornered. However, there are disadvantages that include: design techniques, circuit complexity and size, system speed, and test and evaluation methods (Refs.[8] and [10]).

a) Design Techniques

As discussed, truly asynchronous logic is very difficult to design quickly. Static and dynamic hazards and critical races must be eliminated. Additionally, synchronous logic techniques dominate the circuit design and fabrication field. There are very few CAD systems that support asynchronous design. In fact, many tools have circuit optimization routines that could delete a logic gate required for asynchronous operation, thereby reintroducing a hazard.

Another barrier to asynchronous design is the historic aversion of design engineers to the topic. Academia has stressed for decades the reliability and utility of synchronous design. Engineers have been taught to synchronize asynchronous inputs and build globally clocked machines to handle the required calculations (Refs.[1] and [10]).

b) Circuit Complexity and Size

Truly asynchronous designs are difficult and complex to realize. They routinely require more components and cells to eliminate hazards; therefore, they require more layout area. Two-rail systems are larger and more complex because they require extra logic gates to realize the double signal output. Micropipelines closely resemble synchronous systems in terms of logic gate requirements for the data path; however, the handshake control path could require a larger total layout area than the global clock. The handshake protocol is also more complex than the synchronizing clock method.

c) Speed

Implementation of two-rail systems has actually proven to be slower than similar synchronous systems. Each data item of the two-rail method must be evaluated, probably with a state machine, which takes longer than standard single-rail logic. Micropipelines can be made quicker latency-wise than their synchronous counterparts as long as the handshake control delay is not longer than the data propagation path. However, there will be no overall improvement in the throughput of similar systems. The longest propagation delay path will still determine the throughput of a micropipeline system. Data behind this worst-case path must wait to move down the pipeline until

the acknowledgement or data accepted handshake has been received. The highest rate of input to the system, and therefore the highest throughput, will be determined by the period of this critical path.

d) Test and Evaluation

The bulk of device test equipment is committed to utilizing synchronous inputs to determine sampled results. Asynchronous designs require more complex test and evaluation tools that are not present in the fabrication plants and engineering labs of today. The advantages of asynchronous logic will not come to fruition until a large volume of reliable, tested systems reach the marketplace. (Ref.[10])

C. CHAPTER SUMMARY

Synchronous and asynchronous design techniques have been studied and utilized since the advent of integrated circuit technology. For decades, synchronous methods have dominated in both the academic and commercial design arenas. As IC systems become smaller, faster, and more complex, asynchronous schemes will continue to mature and carve out a larger niche in the integrated circuit market.

THIS PAGE INTENTIONALLY LEFT BLANK

III. MICROPIPELINE HANDSHAKE PROTOCOL

A. HANDSHAKE TIMING

The benefits of asynchronous design can only be realized by performing a detailed analysis of the various timing requirements inherent in the micropipeline protocol. Recall from Equation 2 there are stringent timing parameters that must be maintained for synchronous logic to work properly. Similar equations can be derived for asynchronous micropipeline operation. With these tools, theoretical comparisons are made to synchronous logic techniques.

1. Request Control Timing

Figure 7 depicts a micropipeline stage with the required elements to perform timing analysis. The Muller-C element represents the control logic interface and is the backbone of the four-cycle handshake (Ref.[12]). The request signal from the previous stage is represented by Data Valid In (DVI) and the acknowledge by Data Accepted Out (DAO) from the next stage. The control logic output is Data Valid Out (DVO) to the next stage and DAO to the previous stage. The Data Valid Out signal is also synchronized with a Latch Close (LC) signal to the memory element. During the request cycle, the propagation length of the data path must equal the propagation length of the control path. Taken from the

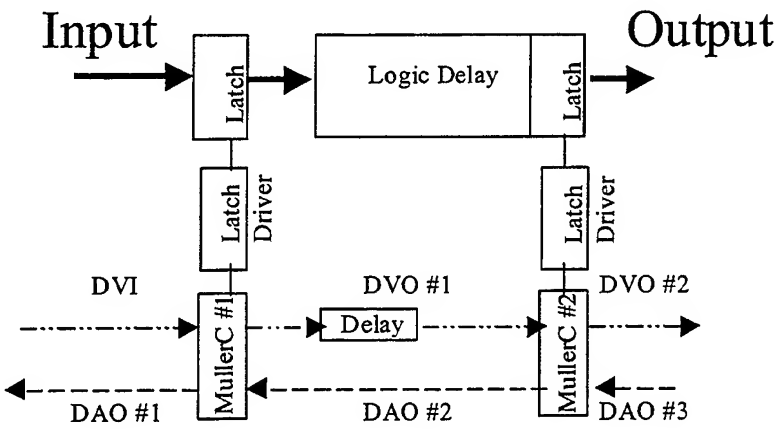

Figure 7. Single Stage Micropipeline

initial request signal at the input to MullerC #1, the data path will display the following dependencies:

$$T_{datapath} \geq T_{MC\#1DVO} + T_{ltchdrv\max} + T_{\log Del} + T_{ltch} + T_{set-up} \qquad \textbf{Equation 3}$$

where

$T_{MC\#1DVO}$ is the propagation delay of MullerC #1 (to DVO),

$T_{ltchdrvmax}$ is the maximum delay of the latch driver,

T_{logDel} is the maximum propagation delay of the path logic,

T_{ltch} is the latch propagation delay, and

T_{set-up} is the latch set-up time.

The control path behaves according to Equation 4:

$$T_{control} \geq T_{MC\#1DVO} + T_{delayelement} + T_{MC\#2DVO} + T_{ltchdrv\min} \qquad \textbf{Equation 4}$$

where

$T_{delayelement}$ is the required control delay,

$T_{1MC\#2DVO}$ is the propagation delay of MullerC #2 (DVO), and

$T_{1tchdrvmin}$ is the minimum propagation delay of the latch driver.

The data path and control propagation equations can be directly compared. Equation 5 depicts the first micropipeline design timing equation.

Equation 5

$$T_{delayelement} \geq (T_{MC\#1DVO} - T_{MC\#1DVO}) + (T_{ltchdrv\max} - T_{ltchdrv\min}) + T_{\log Del} + T_{ltch} + T_{set-up} - T_{MC\#2DVO}$$

Assuming that the latch drivers have been designed to have identical propagation delays, Equation 5 simplifies to:

$$T_{delayelement} \geq T_{\log Del} + T_{ltch} + T_{set-up} - T_{MC\#2DVO} \qquad \textbf{Equation 6}$$

2. Control Loop Timing

Determination of the total period of the control handshake allows for computation of the overall circuit throughput performance. MullerC #1 cannot accept a request signal (DVI) to restart the handshake process until the

following stage has acknowledged receipt of the previous data. Therefore, the worst-case closed loop of the control path will determine the maximum possible system throughput. Equation 7 evaluates the period of one full control loop:

$$T_{contloop} \geq 2*T_{MC\#1DVO(avg)} + 2*T_{delayelement} + 2*T_{MC\#2DAO(avg)} \quad \textbf{Equation 7}$$

where $T_{MC\#1DVO(avg)}$ represents the average DVO propagation delay of MullerC #1 because it goes through both a low-to-high and high-to-low transition. $T_{MC\#2DAO(avg)}$ represents the propagation delay when the DAO signal of MullerC #2 transitions through two handshake cycles. For this micropipeline stage, the delay element has a double impact on the control loop timing period. Implementing the micropipeline of Figure 9 decreases this dependence. The delay element equation now becomes:

$$T_{delayelement} \geq T_{\log Del} + T_{ltch} + T_{set-up} - T_{MC\#1DVO(l-h)} - T_{AND(l-h)} \quad \textbf{Equation 8}$$

where $T_{AND(1-h)}$ is the low to high propagation delay of the AND gate depicted. The control loop equation simplifies to:

$$T_{contloop} \geq 2*T_{MC\#1DVO(avg)} + T_{delelement} + 2*T_{MC\#2DAO(avg)} + 2*T_{AND(avg)} \quad \textbf{Equation 9}$$

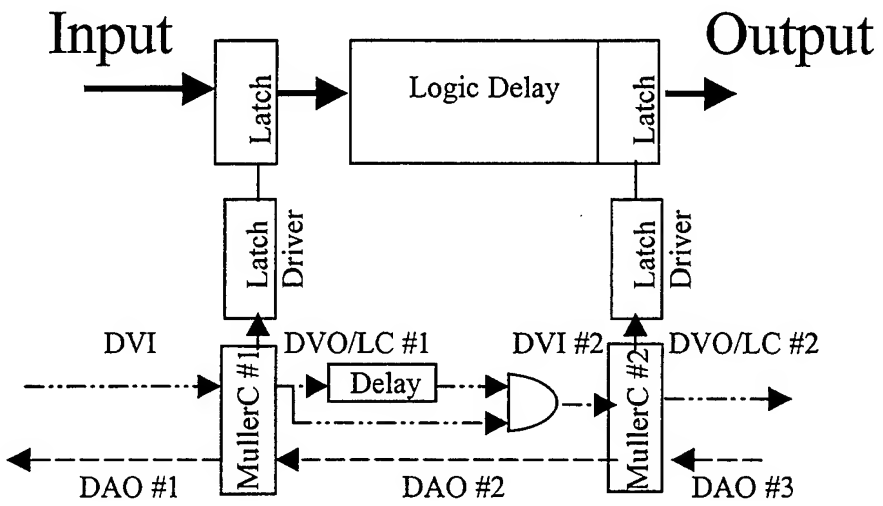

Figure 9. Modified Single Stage Micropipeline

Substitution of Equation 8 into Equation 9 realizes the final equation for the control loop period:

Equation 10

$$T_{contloop} \geq T_{\log Del} + T_{ltch} + T_{set-up} + 2*T_{MC\#2DAOavg} + T_{MC\#1DVO(h-l)} + T_{AND(h-l)}$$

Assuming T_{logDel} represents the worst-case stage delay, Equation 10 determines the period required for throughput calculation of an asynchronous system utilizing the micropipeline technique. Figure 10 details the timing relationships for the four-cycle handshake.

3. Synchronous Logic Comparison

Equation 10 allows for direct theoretical comparison to timing Equation 2 of synchronous techniques. For

asynchronous logic to compete in a throughput sense, Equation 11 must hold.

Equation 11

$$T_{\log Del} + T_{REG} + T_{skew} \geq T_{\log Del} + T_{ltch} + T_{set-up} + 2*T_{MC\#2DAOavg} + T_{MC\#1DVO(h-l)} + T_{AND(h-l)}$$

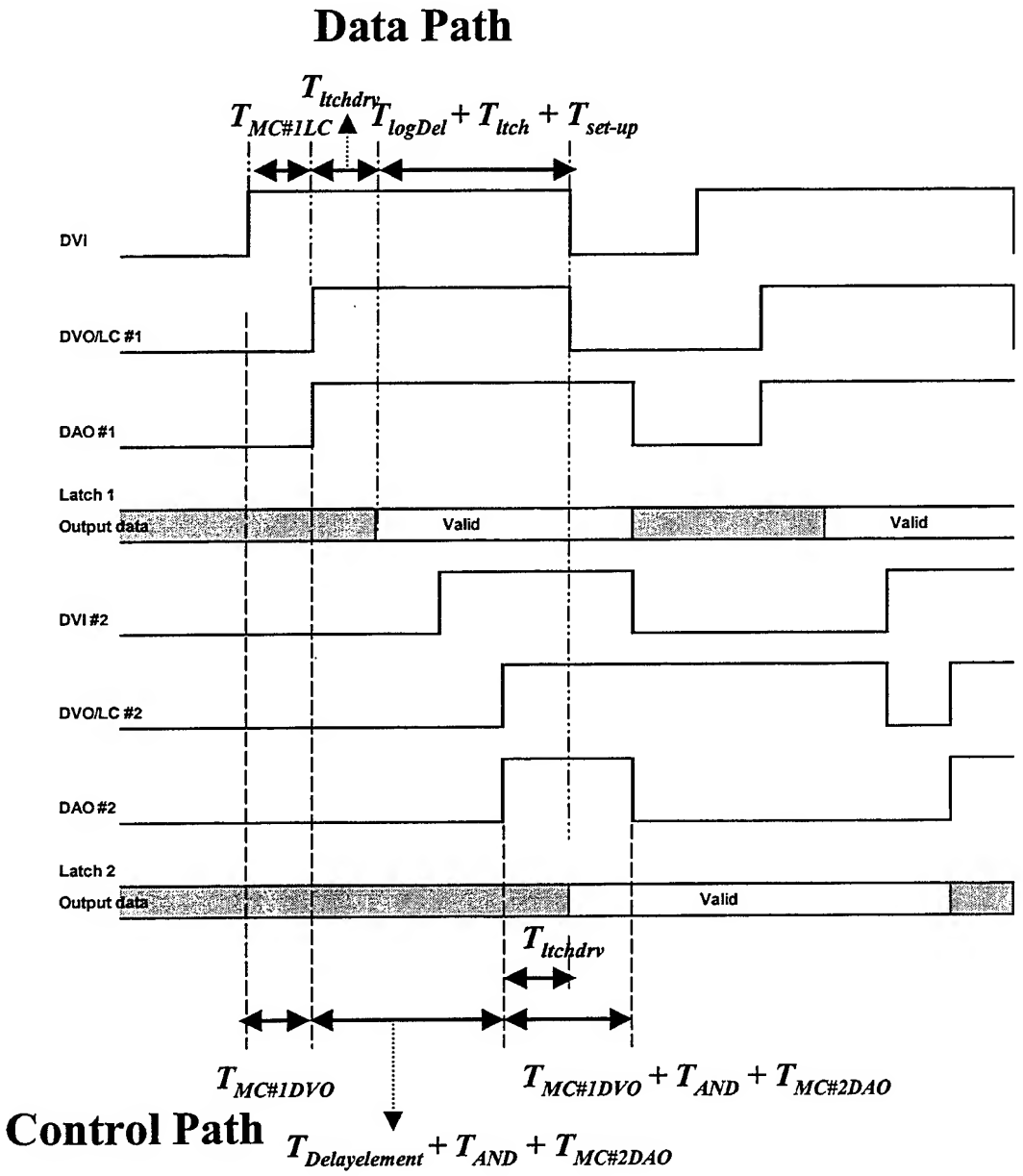

Figure 10. Four Cycle Handshake Timing

Assuming similar logic delays, Equation 11 simplifies to:

$$T_{skew} \geq T_{ltch} + T_{set-up} + 2*T_{MC\#2DAOavg} + T_{MC\#1DVO(h-l)} + T_{AND(h-l)} - T_{REG} \quad \textbf{Equation 12}$$

Asynchronous micropipeline systems operating at peak capability must comply with Equation 12 to derive throughput rate benefits.

B. MULLER-C ELEMENT DESIGN

The Muller-C element is critical to the handshake control of the asynchronous micropipeline. Signal propagation delays through the element determine a variety of system timing relationships; therefore, Muller-C design optimization is necessary.

1. State Machine Development

State machine representations have been developed to implement the four-cycle handshake control.

Muller-C inputs:

DVI: Request from previous stage (Data Valid In)

DAI: Acknowledge from next stage (Data Accepted In)

Muller-C outputs:

DVO: Request to next stage (Data Valid Out)

LC: Memory element control (Latch Closed)

DAO: Acknowledge to previous stage (Data Accepted Out)

Figure 11 is the state transition diagram developed for this handshake protocol utilizing a four state Moore machine approach. Table 2 is the state transition table derived from this state transition diagram.

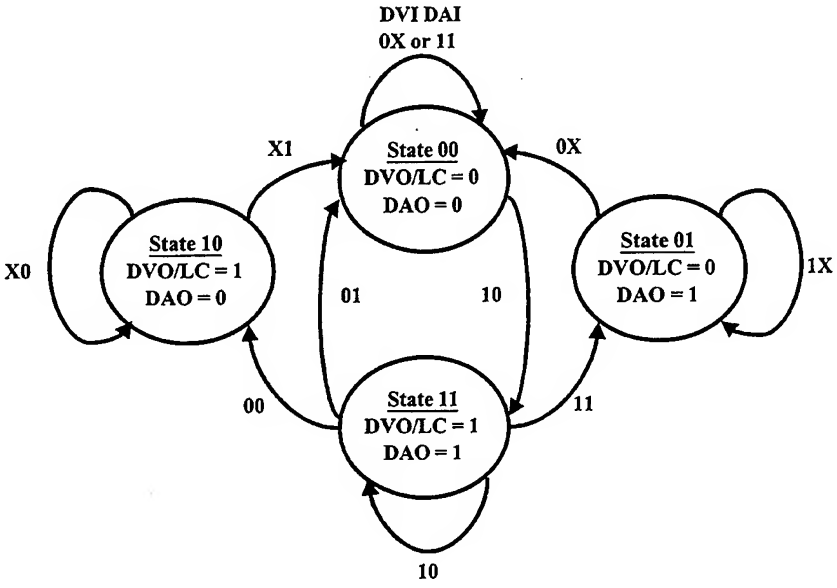

Figure 11. Moore Machine State Transition Diagram

	DVI/DAI				Outputs	
State	00	01	10	11	DVO/LC	DAO
0	0	0	3	0	0	0
1	0	0	1	1	0	1
2	2	0	2	0	1	0
3	2	0	3	1	1	1

Table 2. Moore Machine State Transition Table

2. Logic Design

The Karnaugh map for SR latch implementation of the four state machine is depicted in Figure 12. Current-mode logic (CML) is optimized utilizing OR and NOR gate implementations of digital logic (Ref.[3]). Therefore, the logic gate representation is given by Figure 13.

3. Simulation Results

Tanner SPICE tools were used to simulate the logic

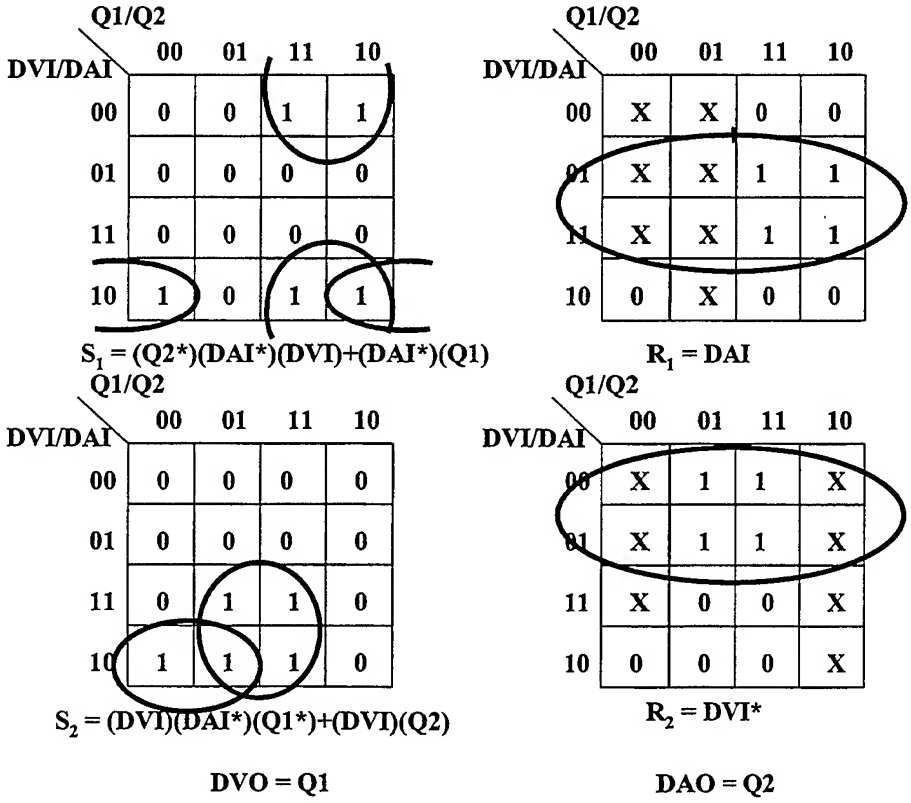

Figure 12. Karnaugh Maps for Muller-C Element Implementation

implementation. For this simulation, the Muller-C element drove a fanout of four logic gates. Figure 14 relates the simulation results and timing relationships.

Table 3 details the propagation delays of selected transitions based on this Muller-C element simulation. As previously discussed, these statistics are important to the design of the 8x8-bit multiplier because they represent delays in the timing control path of the micropipeline.

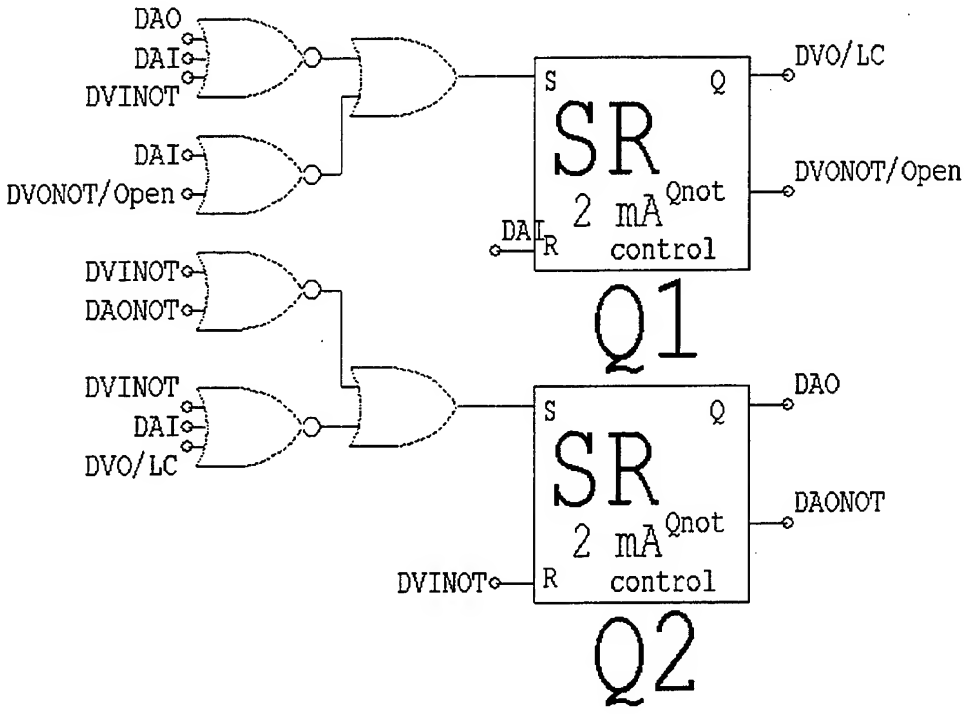

Figure 13. Muller-C Element Logic Implementation

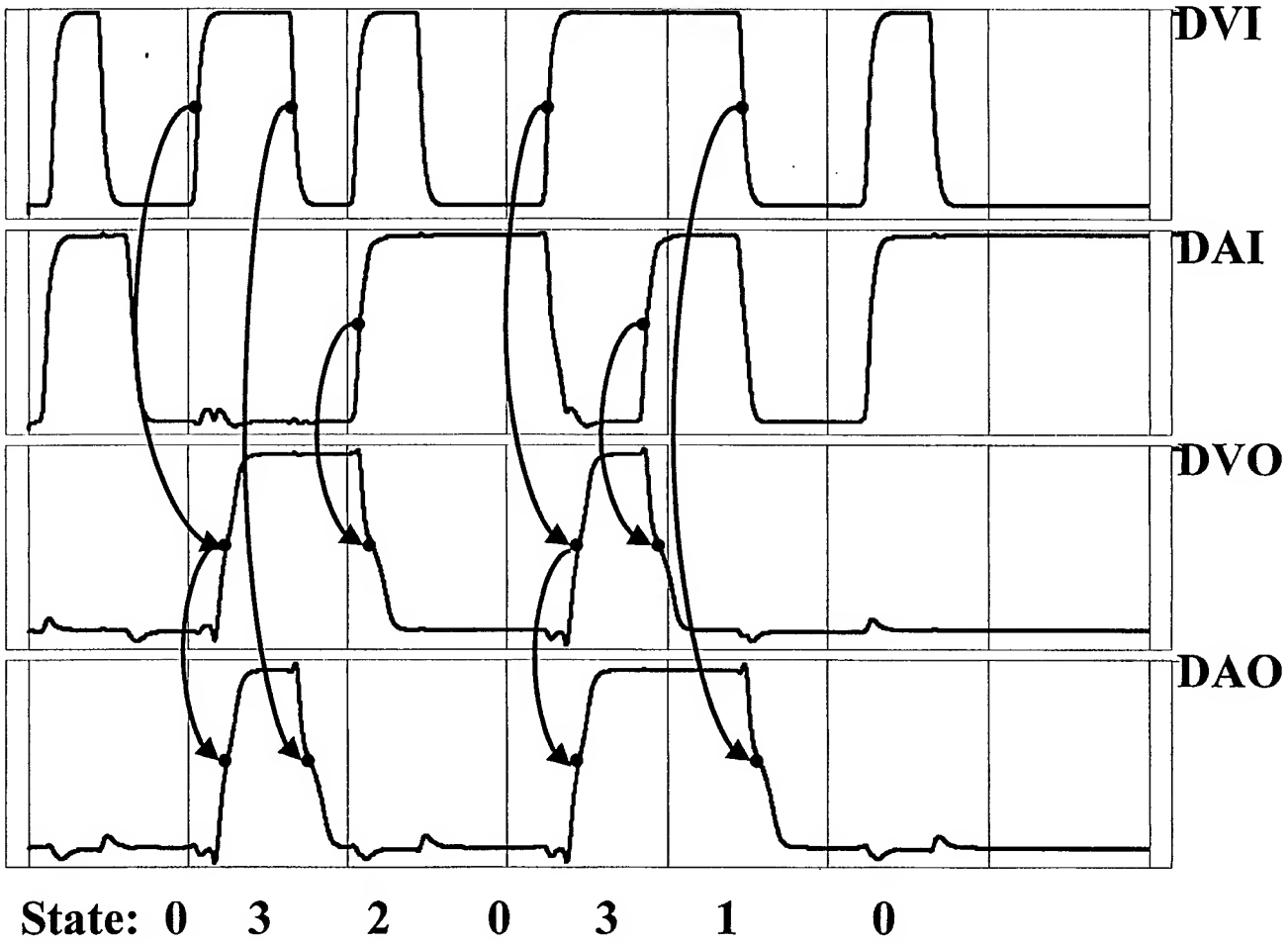

Figure 14. Muller-C Element Simulation Results

Signal	Signal Transition	Propagation Delay
DVO	Low-to-High	94 ps
	High-to-Low	30 ps
DAO	Low-to-High	100 ps
	High-to-Low	40 ps

Table 3. Muller-C Element Propagation Delays

C. SYSTEM SIMULATION TOOLS

Full evaluation of the micropipeline handshake protocol requires simulation testing to insure the Muller-C element behaves appropriately when integrated with memory and digital logic elements. To complete these simulations, key components must be chosen in the InP HBT logic family. These include the basic digital logic, the memory or register elements, and the delay elements critical to the control signal path.

1. Basic Logic

Major John Calvert has designed a set of basic logic gates as the building blocks for the 8x8-bit multiplier. (Ref.[3]). These inverters, buffers, OR, and NOR gates are based on current-mode logic implementations utilizing 0.75 mA bias current and InP HBTs. Two and three-input adders

have also been developed for the circuit from these simple gates.

2. Memory Elements

Synchronous logic utilizing a single phase clock requires D Flip-Flops to provide memory element functions. However, the handshake nature of the asynchronous protocol allows for clocked D latch memory elements. Since D Flip-Flops are typically implemented using two latches in series, a potential 50% memory power reduction is available for an equal number of pipeline stages.

Current-Mode logic lends itself to at least two implementations of a clocked latch, a standard cross-coupled NOR gate method and a current-mode logic latch (Ref.[3]). The CML latch has a shorter propagation delay and requires less power for implementation at the same voltage level. However, it is a more complex circuit and susceptible to noise in its output. Additionally, this unique implementation requires a 2.5 volt process to function while the standard NOR gate method can operate with the basic multiplier logic at 2.0 volts, thus saving power throughout the system. For this research, standard clocked D latches have been selected for the memory elements. The cross-coupled NOR gates are biased at 2.0 mA to decrease propagation delay and allow for a more robust and predictable output signal. TSPICE simulation convergence and

completion are inconsistent below this bias threshold. Table 4 summarizes the critical parameters of the two approaches.

	Synchronous	Asynchronous
Memory Element	D Flip-Flop	D Latch
Voltage	2.5 v	2.0 v
Propagation Delay		
Low-to-High	43 ps	31 ps
High-to-Low	54 ps	5 ps
Set-up time	35 ps	42 ps
Max Total Delay	89 ps	73 ps
Current	7.0 mA	7.3 mA

Table 4. Memory Element Comparison

3. Delay Elements

The control path delay elements are modeled with lossless transmission lines available in the standard TSPICE library. As illustrated in Figure 15, these transmission lines are buffered on both ends to reduce the negative impact the lossless line has on its input signal and produce a consistent, predictable output. It is possible to produce nearly lossless transmission lines of the lengths needed for this research on an integrated circuit; therefore, this

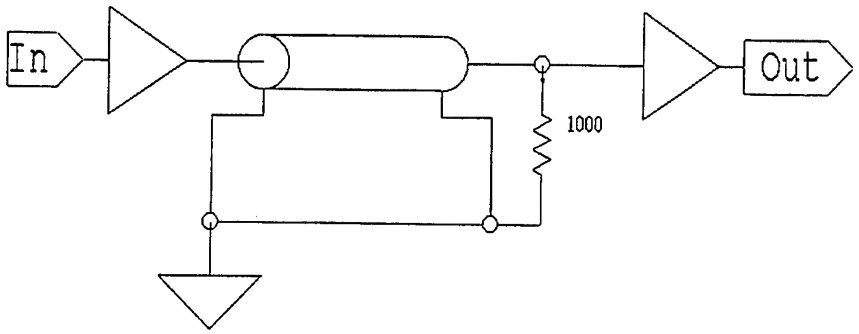

Figure 15. Delay Element Circuit Diagram

technique lends itself to real-world layout and implementation as well.

D. MICROPIPELINE SIMULATIONS

Single-stage and three-stage simulations were developed to verify system operation with the Muller-C element providing the handshake protocol. The initial Data Valid In signal was timed to insure that the input latch received and latched data just as set-up time was achieved to insure the timing criteria set forth earlier have been strictly met.

1. Delay Element Redesign

The three-stage simulation highlighted a problem with the integration between the delay element and the proper data valid input signal to the next stage. For stages with significant propagation delays, it is possible for a new Data Valid Out signal to arrive at the AND gate before the previous delayed signal has cleared the logic. This

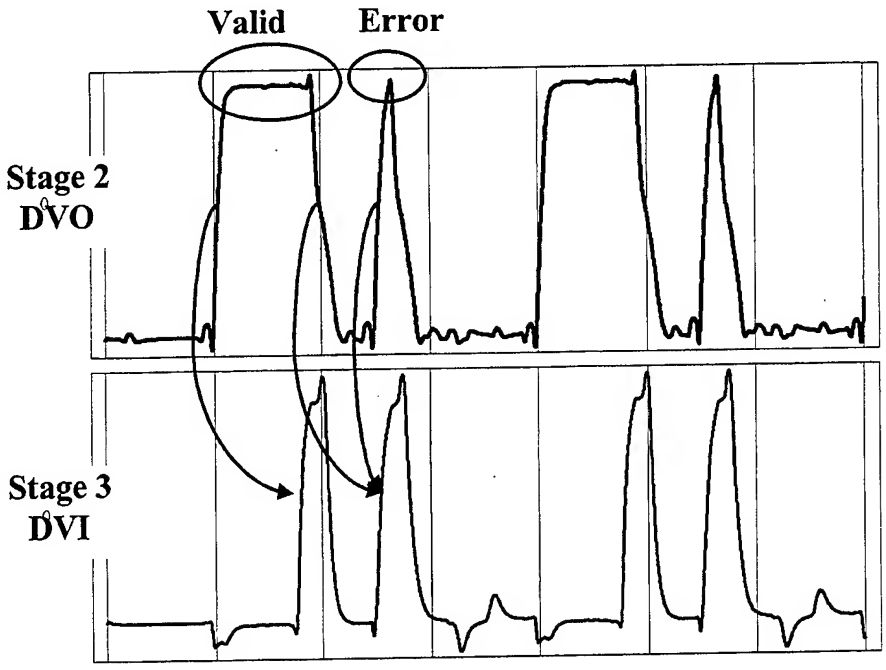

Figure 16. Handshake Timing Results

Delay Element Anomaly

situation provides a false Data Valid In (DVI) signal to the next stage. Figure 16 depicts these simulation results and highlights the phenomenon.

The delay element has been redesigned to produce a pulse on the delayed signal line (Figure 17). The concept was verified in follow-on simulations.

2. Circuit Failure

There are two fundamental failure modes for the timing relationships in the asynchronous micropipeline model. If

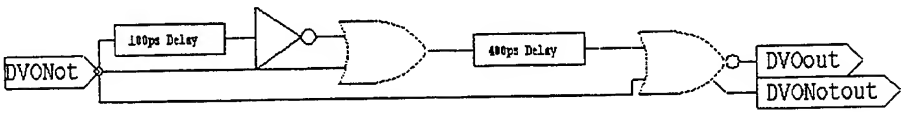

Figure 17. Pulsed Delay Element

the propagation of the control path delay element is too short, the data into the next stage will not latch appropriately, as it will not achieve the set-up time for the memory elements. Test results are quick and easy to verify assuming the input latch just meets set-up time.

When the period of the control path is underestimated, the Data Valid signal into the longest control path will fire before the Data Accepted signal from the next stage has been fully received. In time, this will drive the timing relationships out of limits and produce improper system outputs. Figure 18 graphically represents the results of an underestimated control path period.

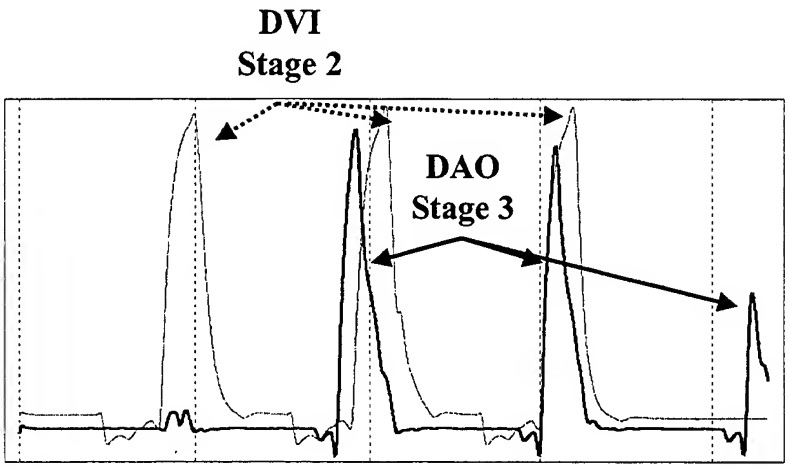

Figure 18. Handshake Breakdown-Control Period

3. Tabulated Results

Table 5 provides results for the one stage and three stage simulations. The data clearly shows that the timing equations derived earlier in this research are valid for the micropipeline handshake technique.

	One Stage	Stage One	Stage Two	Stage Three
#1 Latch Drive Max	0 ps	0 ps	0 ps	0 ps
#2 Latch Drive Min	0 ps	0 ps	0 ps	0 ps
Logic Delay	200 ps	200 ps	400 ps	100 ps
$T_{latch} + T_{set-up}$	73 ps	73 ps	73 ps	73 ps
$T_{MCDVO(1-h)}$	94 ps	94 ps	94 ps	94 ps
$T_{AND(1-h)}$	24 ps	24 ps	24 ps	24 ps
$T_{delayelment}$ theoretical (Equation 8)	155 ps	155 ps	355 ps	55 ps
$T_{delayelement}$ actual	160 ps	160 ps	370 ps	60 ps
$2*T_{MCDAO(avg)}$	140 ps	140 ps	140 ps	140 ps
$2*T_{MCDVO(avg)}$	120 ps	120 ps	120 ps	120 ps
$2*T_{AND(avg)}$	40 ps	40 ps	40 ps	40 ps
$T_{contloop}$ theoretical (Equation 9)	455 ps	455 ps	*655 ps*	355 ps
$T_{contloop}$ actual	450 ps	650 ps	650 ps	650 ps

Table 5. Micropipeline Simulation results

E. CHAPTER SUMMARY

The four-cycle micropipeline handshake is a complex mechanism used to provide local timing signals for an asynchronous logic system. The Muller-C element provides the state machine functions that control the handshake. By following the inherent timing relationships between the logic and control paths of the micropipeline, system engineers can design operable high-throughput integrated circuits.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. MULTIPLIER DESIGN AND PERFORMANCE ANALYSIS

A. MULTIPLIER DESIGN

1. Multiplier Implementations

There are numerous methodologies utilized to implement binary multipliers. These include multiply-accumulators and array multipliers.

a) Multiply-Accumulator

The multiply-accumulator is a relatively simple multiplier implementation. Assuming the multiplier bit is a logical one, the multiplicand is shifted according to the multiplier bit position and added to the previous result. The multiply-accumulate method is illustrated in Figure 19 in both decimal and binary formats. While the multiply-accumulator is simple to design, it requires numerous ripple-carry addition manipulations which take time to process. The higher-order bits must wait in turn until the lower order carry bits have been determined through a series of binary adders. Therefore, this implementation inserts unnecessary delay into the system.

b) Array Multiplier and Carry-Save Addition

For this research, an array multiplier has been designed that takes advantage of available parallelism and optimizes memory element requirements and system modularity.

```
  123        Multiplicand    Multiplier    Accumulator
   27
 3321          123              7          1 1
                                            861   carry
               123              2           246 ← shift
                                           3321

 1111011      Multiplicand    Multiplier     Accumulator
    11011
1100111111001  1111011          1          1111111   ripple carry
                                              1111011
               1111011          1          1111011     ← shift
                                           101110001
               1111011          0          0000000     ←shift
                                           1111110     ripple carry
                                           101110001
               1111011          1          1111011     ←shift
                                           1110000     ripple carry
                                          10101001001
               1111011          1          1111011     ←shift
                                          1100111111001
```

Figure 19. Multiply-Accumulator Model

The implementation is a hybrid of the multiply-accumulate method and a large array multiplier.

Array multipliers can be designed with the entire multiplier array realized as the first step of the pipeline with follow-on stages reducing the operands accordingly (Ref.[1]). This method could require as many as 64 (8x8) memory elements in the first stage alone which is excessive when power considerations are taken into account. However, the full array multiplier can be reduced in the most efficient manner through a proper carry-save and carry-completion adder chain.

Carry-Save addition involves taking the full row of carry bits as an operand into the next adder stage. The multiplier operand array is then reduced quickly to the

final ripple-carry adder stages. Figure 20 relates the approach.

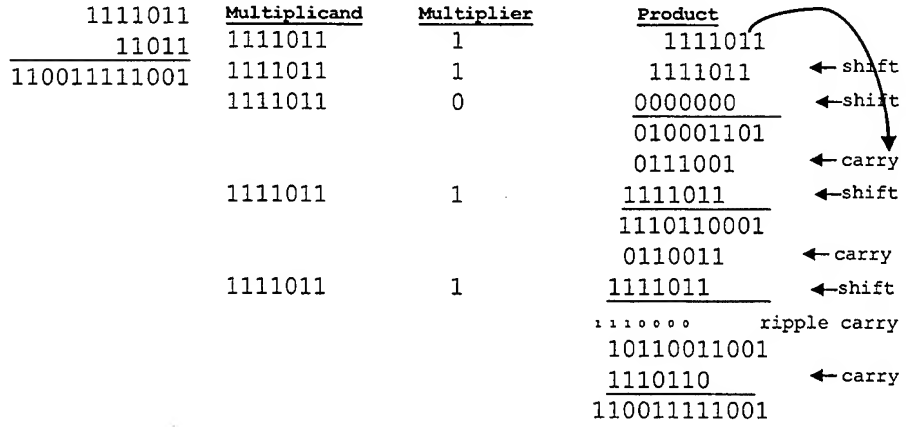

Figure 20. Carry-Save Addition Model

2. System Modularity

Figure 20 is a relatively accurate illustration of the modularity of the multiplier implementation utilized in this research. The first stage is a Carry-Save Adder (CSA) that manipulates the eight multiplicand bits and the three least significant multiplier bits. The carry-save adder is a system of inverters, buffers, NOR gates and adders that provides a row of eight sum bits and a row of eight carry bits to the next stage. The next five stages of the multiplier manipulate the multiplicand and the remaining five multiplier bits in order. The final eight stages of the system provide a series of Carry-Completion Adders (CCA) to realize the final 16-bit product (Figure 21).

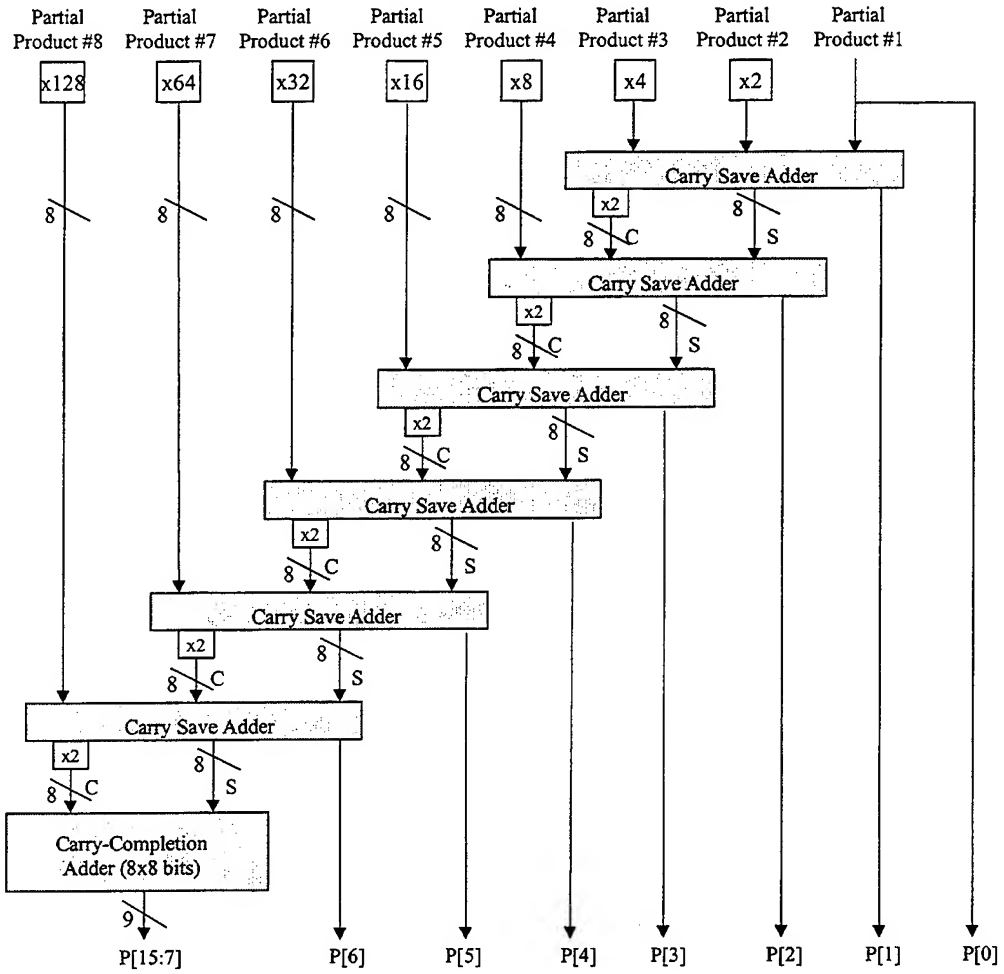

Figure 21. 8x8-bit Multiplier Schematic

(From Ref.[3])

In order to pipeline the multiplier, registers are placed between selected Carry-Save or Carry-Completion Adders. Figure 22 is an example of a four stage pipeline utilizing these modular units. A more in-depth perspective into specific system design of the multiplier can be found in Major John Calvert's Naval Postgraduate School Master's

thesis which investigates a synchronous implementation of this 8x8-bit system (Ref.[3]).

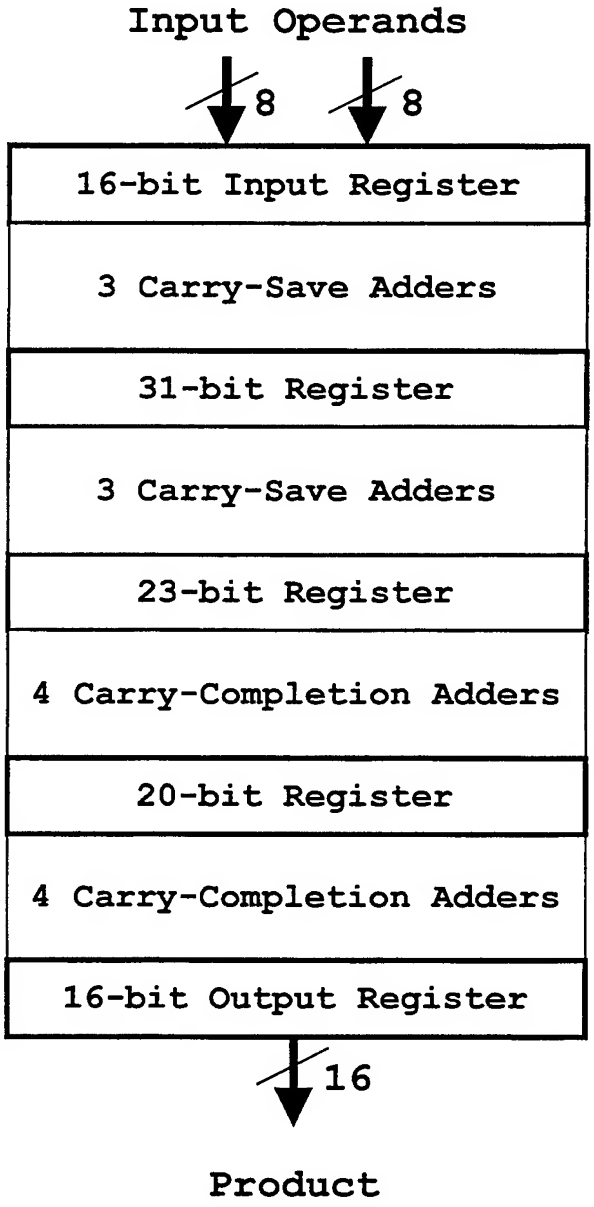

Figure 22. Four Stage Pipeline Example

3. Critical Path Determination

To fully optimize the asynchronous pipelined multiplier, it is imperative that the critical path of each pipeline stage be determined. Critical path determination involves finding the maximum propagation delay from start to finish of a micropipeline stage. In a synchronous system, the requirement is to find only the single worst-case logic delay path for the entire multiplier implementation. For a micropipelined system, the critical path and propagation delay must be determined for each stage.

Intuitively, it would seem that the critical path of a pipeline stage will simply be the summation of the maximum propagation delays of each of the series elements in the particular stage. For the ten stage pipelined implementation of this research, this concept holds relatively true. However, the circuit designer must also consider possible dynamic logic hazards when, for example, a stage output transitions from high-to-low and back to high during one request-acknowledge period. Figure 22 details an example of one such potential occurrence for the second Carry-Save Adder of the 8x8-bit multiplier. If the operands into the NOR gate have been delayed long enough for the adder to produce the logical one on the output before the NOR gate output stabilizes, the adder will actually transition from low-to-high and back to low. Without considering these

timing implications, a system designer could easily mistake the propagation delay for this sum bit as zero because it starts and finishes in the low state. However, a more in-depth analysis reveals that there are two transitions which must be considered for the pipeline stage delay on this sum bit output.

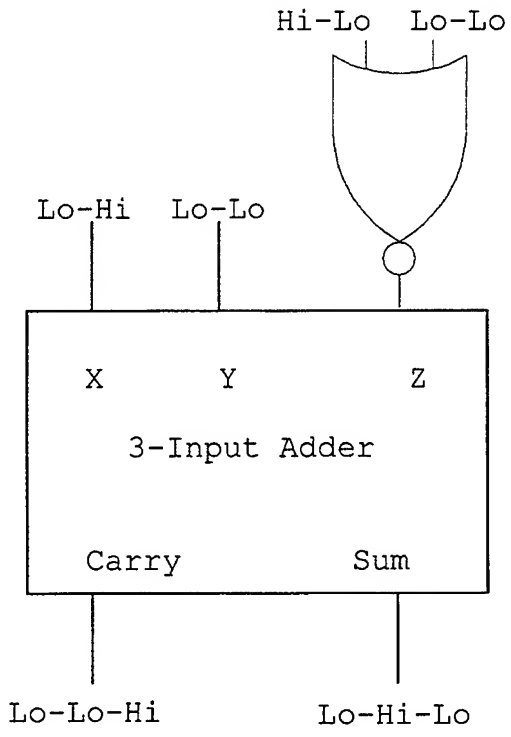

Figure 22. Sum bit Critical Path Dynamic Hazard

As previously stated, the summation of the longest propagation delays for the series elements in a thinly sliced pipeline will allow for a reasonable estimation of the length of the critical path for that stage. However, for longer pipeline stages and for the single stage 8x8-bit

multiplier, these system hazards make accurate determination of the critical path extremely difficult. Without this knowledge, the system can not be efficiently designed or tested as all possible operand combinations must be considered. Intuitively, it might seem that 255*255 $(2^8-1)^2$ would provide the operands for the worst case system path as the largest 16-bit product will be realized. It is important to consider that the most-likely worst-case operands will, at a minimum, produce a complete ripple-carry through each of the final Carry-Completion Adders in a non-pipelined system. The intuitive case does not provide for this eventuality.

For this research, a C++ program was developed which evaluates 256^2 separate operand pairs while resetting the system to the all zero case between each test. The program time steps the multiplier in one pico-second increments and accurately predicts the occurrence of dynamic hazards and highlights potential worst-case operands. This tool was utilized to determine the operands which drive the critical path for each pipelined implementation of the multiplier. Tanner SPICE simulations of various sets of these operands indicate that likely critical paths have been identified. Additionally, the synchronous and asynchronous evaluations of the multiplier both utilized the same operand pairs to maintain an accurate comparison.

Figure 23 is an example of the ripple-carry effect which drives the critical path length of a single stage micropipeline of the 8x8-bit multiplier. The darkest waveform represents P[10], the tenth product bit, which transitions from low-to-high-to-low two times during the simulation of one set of operands. Additionally, higher order bits also make transitions from low-to-high-to-low before the system has actually produced a stable output. The operand pair that drives this scenario was predicted by the C++ program.

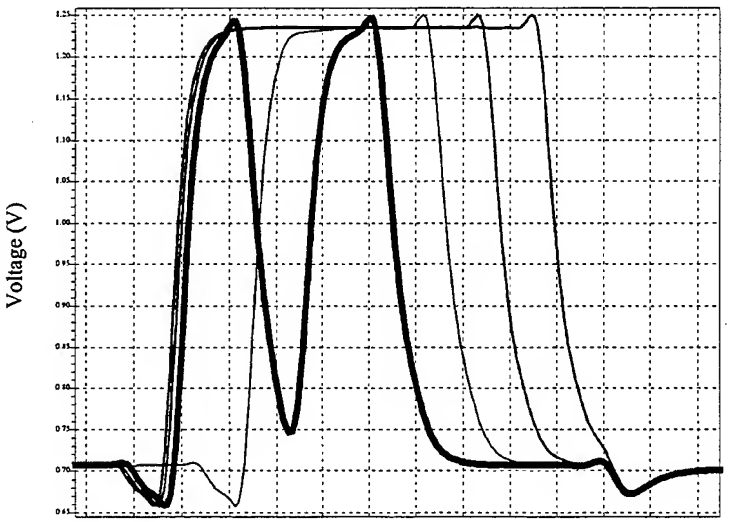

Figure 23. Ripple-Carry Example for Critical Path Determination

B. SIMULATION RESULTS AND PERFORMANCE ANALYSIS

Five separate implementations of the micropipelined 8x8-bit multiplier were developed and simulated using Tanner SPICE tools. The steps in the development and test process included determination of the logic delay between each stage followed by multiple simulations of each micropipeline stage to determine the delay element requirements. Finally, these timing values along with voltage and current data for each implementation were analyzed to determine the optimum multiplier configuration. The metrics used include system latency, throughput, power consumption, speed-power ratio, and device count.

1. Simulation Results

a) Timing Relationships

The logic for each pipeline stage of the multiplier was simulated to determine the basic logic delay involved. Input and output operands for each pipeline stage were chosen based on those predicted by the C++ program previously discussed. These sum and carry operands form the basis of the test data for the bulk of the micropipeline simulations.

After logic simulation, the equations developed in Chapter III were utilized to determine the requirements for the delay elements between each stage. Lossless transmission lines were utilized to implement the delay elements.

Adequate bracketing of the required delays was accomplished to insure optimum system performance. Figure 24 illustrates both a valid and invalid test of a micropipeline stage. In this case, the critical output bit for the stage fails to achieve latch set-up time in the second simulation. Systematic evaluation of all five multiplier implementations realized delay element requirements for further timing relationship analysis. Additionally, the control period

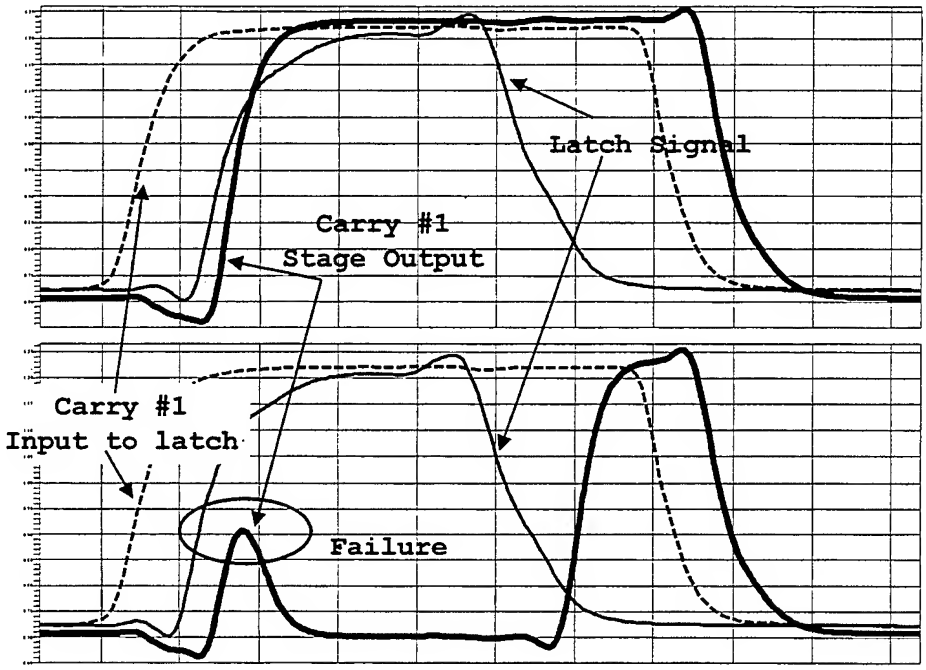

Figure 24. System Performance Bracket

relationships were estimated and evaluated for the worst-case path for each multiplier. These tests determine system throughput as previously discussed.

b) System Current Requirements

Separate voltage sources were designated in the Tanner SPICE schematics so that current requirements could be determined for the different functions inherent in the system. Each gate has either a logic element, control element, or memory element DC voltage source of 2.0 volts. Graphical results of the current used by these voltage sources allow for quick power consumption calcualtions. Tables 6 and 7 detail the timing and power requirements for each stage of the separate multiplier implementations.

	Multiplier Implementation						
	One Stage	Two Stage		Four Stage			
Stage	1	1	2	1	2	3	4
Latch Drive Max (ps)	40	40	41	40	41	41	40
Next Latch Drive Min (ps)	40	39	40	38	39	38	40
Logic Delay (ps)	790	380	420	230	230	200	240
Delay Element theoretical (ps)	755	346	386	197	197	168	205
Transmission Line theoretical (ps)	725	316	356	167	167	138	175
Pass	730	325	375	185	185	150	200
Min Period (ps)							
Theoretical	1045	676			495		
Pass	1050	680			510		
Throughput (x10^{9}MPS)							
Theoretical	0.96	1.48			2.02		
Actual	***0.95***	***1.47***			***1.96***		
Latency (ps)	870	960			1220		
Logic Current	1101	1101			1101		
Memory Current	353	586			1095		
Control Current	114	186			340		
Total Current (mA)	1568	1873			2536		
Logic Power	2202	2202			2202		
Memory Power	706	1172			2190		
Control Power	228	372			680		
Total Power (mW)	***3136***	***3746***			***5072***		

Table 6. Timing and Power Data for One, Two, and Four Stage Implementations

Multiplier Implementation													
	Six Stage						Ten Stage						
Stage	1	2	3	4	5	6	1	2-5	6	7	8	9	10
Latch Drive Max (ps)	40	41	41	41	41	38	40	41	41	41	38	40	38
Next Latch Drive Min (ps)	38	38	39	38	38	40	38	38	39	36	40	38	40
Logic Delay (ps)	190	160	160	130	190	120	120	100	100	60	120	120	120
Delay Element theoretical (ps)	157	128	127	98	158	83	87	68	67	30	83	87	83
Transmission Line theoretical	127	98	97	68	128	63	67	48	47	10	63	67	63
Pass	140	110	110	80	140	80	80	60	60	20	80	80	80
Min Period (ps)													
Theoretical			448							377			
Pass			450							400			
Throughput (x10 ^{9}MPS)													
Theoretical			2.23							2.65			
Actual			***2.22***							***2.50***			
Latency (ps)			1430							1740			
Logic Current			1101							1101			
Memory Current			1599							2731			
Control Current			495							802			
Total Current (mA)			3195							4634			
Logic Power			2202							2202			
Memory Power			3198							5462			
Control Power			990							1604			
Total Power (mW)			***6390***							***9268***			

Table 7. Timing and Power Data for Six and Ten Stage Implementations

2. Multiplier Performance Analysis

The previous tables supply the bulk of the data required to make performance evaluations of the micropipelined 8x8-bit multiplier. Graphical representations have been developed to relay selected performance metrics.

a) Latency

The latency of the system is defined as the time it takes the first set of operands to transition through the entire multiplier and provide a stable product out of the output memory register. As the number of pipeline stages increases, latency performance degrades due to the propagation delay of the added memory register banks in the logic path.

b) Throughput

System throughput is a measure of the rate at which operands can be fed into the system and still realize accurate output. The throughput of the micropipelined system is bounded by the handshake control period as discussed in Chapter III. Latency and throughput for the separate implementations of the asynchronous 8x8-bit multiplier are graphically represented in Figure 25. For this research, the throughput units will be in multiplies-per-second (MPS).

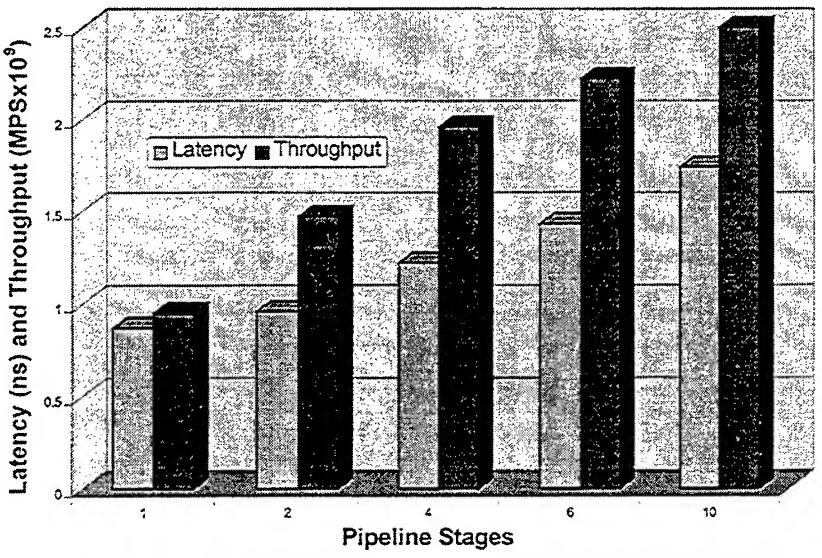

Figure 25. Micropipeline Latency and Throughput

c) Power Consumption

In order to relate the relative contributions of the logic, memory, and control elements, a multi-dimensional bar graph is utilized to depict system power consumption (Figure 26). As expected, analysis indicates that as the number of pipeline stages increases, the memory elements consume the bulk of the system power. As an example, for the two stage system, the logic consumes 60% of the total power, the control elements 10% and the memory devices 30%. When pipelined to ten stages, these elements consume 24%, 17% and 59% respectively.

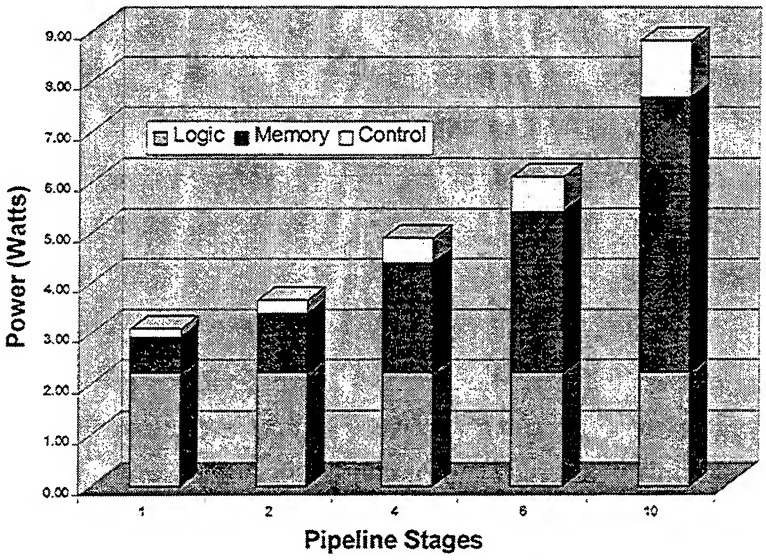

Figure 27. Micropipeline Power Consumption

d) Speed-Power Ratio

The speed-power ratio is a good metric of system efficiency that relates system throughput to total power consumed. The maximum point of the speed-power ratio curve indicates the optimal system pipeline configuration. The negative aspect of increased power consumption is balanced against the benefit of higher system throughput or device speed. The graph of Figure 28 indicates that optimum efficiency of the asynchronous 8x8-bit multiplier lies between the two and four stage implementations.

e) Device Count

The number of transistors and resistors in an integrated circuit relates directly to the area and complexity of the layout required to fabricate the system

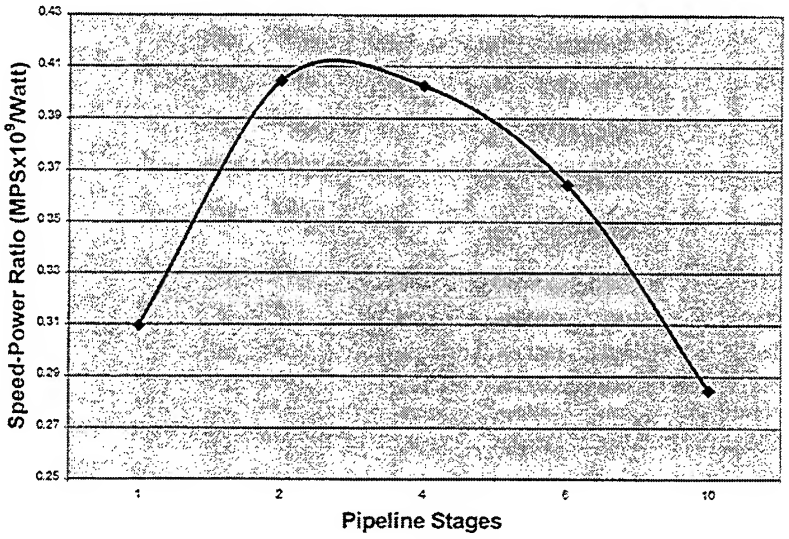

Figure 28. Micropipeline Speed-Power Ratio

(Ref.[14]). The logic, memory, and control devices of the different stage implementations of the 8x8-bit multiplier have been tabulated and are presented graphically in Figure 29. Similar to the power consumption results, as the number of pipeline stages increases, a large percentage of the devices required for fabrication fall into the memory element category.

C. CHAPTER SUMMARY

An 8x8-bit asynchronous multiplier has been designed and systematically tested through a range of input operands in a computer simulation environment. The results of these simulations indicate that the implementation shows promise in terms of overall system power consumption. However, the time lost in the four-cycle handshake protocol limits system

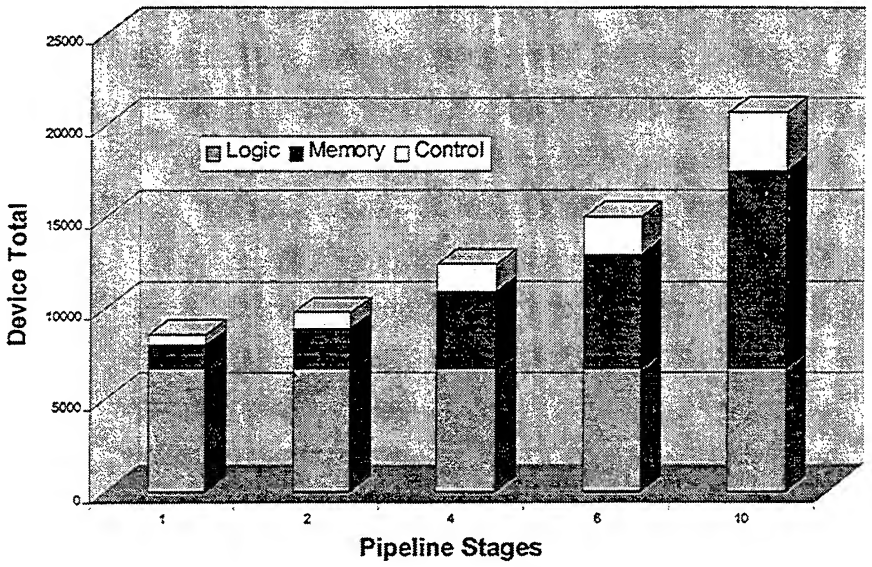

Figure 29. Micropipeline Device Count

throughput. The next logical step, comparison to an optimum design of a synchronous multiplier, is completed in the following chapter.

V. COMPARISON TO SYNCHRONOUS IMPLEMENTATION

A. PERFORMANCE ANALYSIS

Major Calvert has designed a synchronous implementation of the 8x8-bit pipelined multiplier using similar logic configurations to the multiplier disussed in the previous chapter (Ref.[3]). The primary differences between the two approaches lies in the choice of memory elements and in the method of control and synchronization. For the synchronous implementation, the Current-mode Latch forms the basis for D Flip-Flops utilized in the register banks. These latches have superior propagation delay properties; however, they require 2.5 volts to operate. Additionally, the clock drivers in the synchronous system require a 2.0 mA bias current to function effectively. For the asynchronous system, the cross-coupled NOR gate latch can operate with 2.0 volts and 0.75 mA bias current drivers. Because the asynchronous system can utilize latches vice Flip-Flops as memory elements, the loss in propagation delay
for the multiplier logic path is generally not realized. However, the throughput of the asynchronous system is governed by the propagation delay of the Muller-C elements, which is extensive and effected only by standard logic propagation properties.

For the comparison between the two types of implementations, the metrics used include system latency, throughput, power consumption, speed-power ratio, and device count. Table 8 summarizes the critical data.

1. Latency

Analysis of the data from table 8 indicates there are no real trends for latency performance between the two implementations. Theoretically, the asynchronous technique should display a shorter latency period; however, numerous factors mitigate this advantage. First, minor logic implementation differences between the two multiplier approaches allow for slightly different system logic delays. The primary reason the latency results are inconclusive is that the synchronous and asynchronous multipliers are sliced relatively equally between the stages for most of the implementations. The asynchronous method has a shorter latency period when there is a large disparity between the longest and shortest stage delay paths. When they are essentially equal, no improvement in performance is realized.

	Multiplier Implementations									
	One Stage		*Two Stage*		*Four Stage*		*Six Stage*		*Ten Stage*	
	Synch	Asynch	Synch	Asynch	Synch	Asynch	Synch	Asynch	Synch	Asynch
Latency (ps)	750	870	1000	960	1160	1220	1380	1430	1800	1740
Throughput (x10^9MPS)	1.33	0.95	2.00	1.47	3.45	1.96	4.35	2.22	5.56	2.50
Current (mA)										
Logic	1273	1101	1273	1101	1273	1101	1273	1101	1273	1101
Memory	315	353	529	586	1015	1095	1487	1599	2574	2731
Clock/Control	186	114	358	186	686	340	1030	495	1745	802
Total	1773	1568	2159	1873	2974	2536	3790	3195	5591	4634
Power (W)										
Logic	3.18	2.20	3.18	2.20	3.18	2.20	3.18	2.20	3.18	2.20
Memory	0.79	0.71	1.32	1.17	2.54	2.19	3.72	3.20	6.44	5.46
Clock/Control	0.46	0.23	0.89	0.37	1.72	0.68	2.57	0.99	4.36	1.60
Total	4.43	3.14	5.40	3.75	7.44	5.07	9.47	6.39	13.98	9.27
Speed-Power Ratio (GHz/W)	0.30	0.30	0.37	0.39	0.46	0.39	0.46	0.35	0.40	0.27
Device Count										
Logic	6304	6672	6304	6672	6304	6672	6304	6672	6304	6672
Memory	704	1280	1210	2200	2332	4240	3432	6240	5940	10800
Clock/Control	231	585	418	877	803	1490	1188	2103	2046	3204
Total	7239	8537	7932	9749	9439	12402	10924	15015	14290	20676

Table 8. Performance Data for Synchronous vs. Asynchronous Multiplier Comparison

2. Throughput

Figure 30 relates the difference between the synchronous and asynchronous approaches in terms of system throughput. As expected, the synchronous system far outperforms the asynchronous implementation for this metric; especially when the multiplier is heavily pipelined. The time delay inherent in the four-cycle handshake limits the speed performance of the asynchronous method. The throughput of the asynchronous method does increase linearly; however, not at the pace that the synchronous method increases.

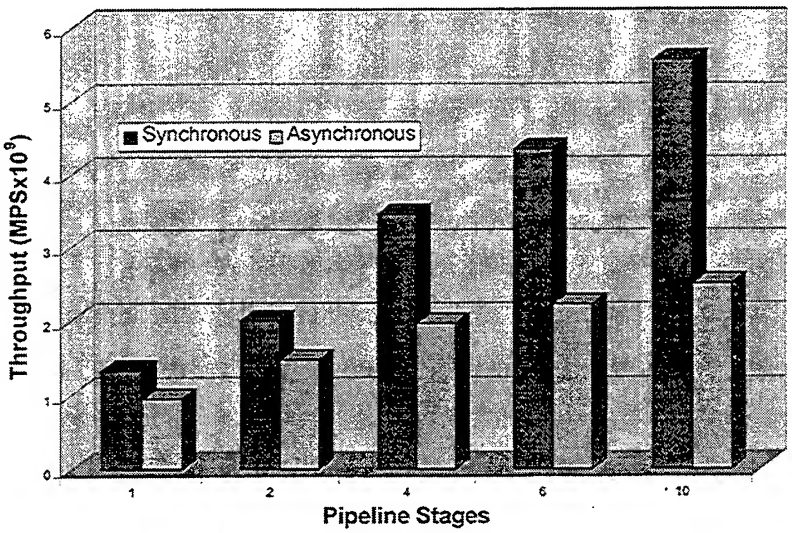

Figure 31. Throughput Comparison

3. Power Consumption

The asynchronous multiplier implemented in this research demonstrates superior performance in terms of

system power consumption (Figure 31). The 2.0 volt versus the 2.5 volt power rail determines a large portion of the power savings. The relative current loads drawn by the elements in the system also play a role in the power consumption differential. The logic and memory elements maintain a relatively consistent relationship across the different stage implementations; however, the requirement for 2.0 mA clock drivers for the synchronous system realizes a higher current load overall. For instance, the control elements of the asynchronous one-stage multiplier require 39% less current than the clock drivers of the synchronous system. In the ten-stage implementation, it is a 54% difference. This comparison highlights the theoretical power savings predicted for the asynchronous system.

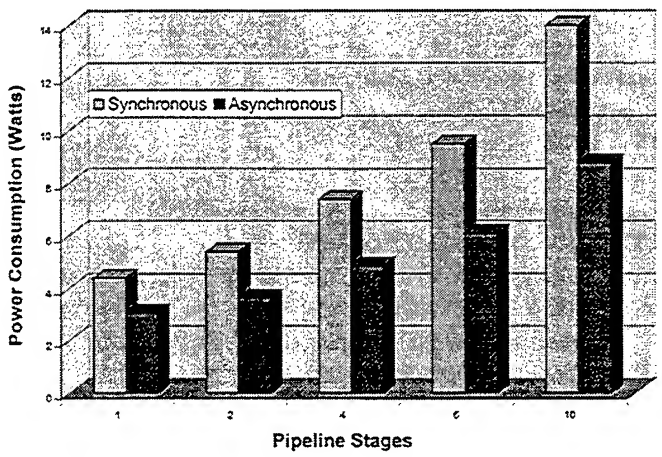

Figure 31. Power Consumption Comparison

4. Speed-Power Ratio

The speed-power ratio displayed by the synchronous and asynchronous multiplier implementations are depicted in Figure 32. This graph includes representations of the synchronous implementations over several values of theoretical clock skew. For multipliers with low levels of pipelining, the depth of the global clock distribution tree is minimum; therefore, the impact of skew is not as prevalent for the one and two-stage implementations. As the number of pipeline stages increases, these skew values

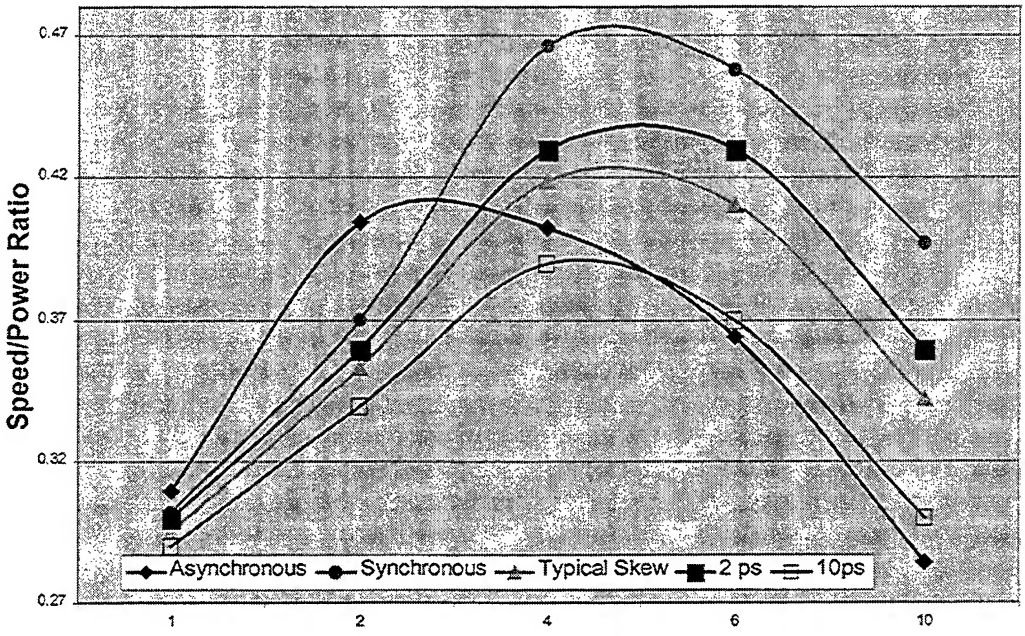

Figure 32. Speed-Power Ratio Comparison

consume more of the minimum clock period required with a corresponding decrease in system speed-power ratio. The typical skew value as depicted was calculated by approximating device induced skew as 20% of the worst-case propagation delay for the synchronous clock driver circuit. For the ideal synchronous implementation, the high throughput rate attainable significantly outweighs the power advantage gained by the asynchronous technique. However, when the effect of clock skew is added to the comparison, the peak speed-power ratio of the asynchronous multiplier begins to approach that of the synchronous design. (Ref[3]).

5. Device Count

As previously discussed, the device count metric displays relative chip layout area for a given integrated system. Figure 33 indicates that the synchronous system

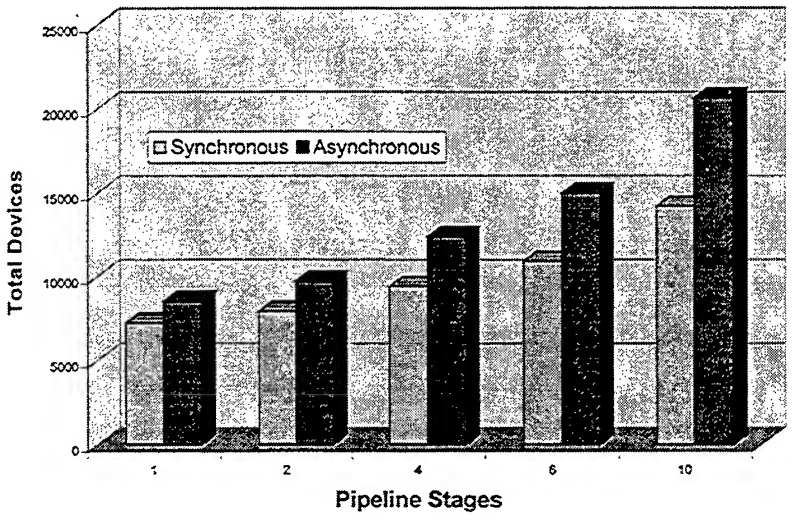

Figure 33. Device Count Comparison

would require a smaller chip area and may predict a higher wafer yield for system fabrication. Smaller individual chip area would equate to more systems on a single silicon wafer, which is more cost effective (Ref.[14]). The largest contributing factor to increased chip area for the asynchronous technique lies in the choice of memory elements. The D Flip-Flop requires 20 transistors and 10 resistors to produce while the cross-coupled NOR gate latch requires 24 transistors and 16 resistors to implement. The synchronous logic therefore requires 10 fewer devices per element which is extensive in the 31 element register banks required in the ten-stage pipeline, for instance.

6. Non-Logic Propagation Delay

Figure 34 graphically illustrates the primary performance differential between the synchronous and asynchronous approaches. When taken as a percentage of the throughput period (1/throughput), the cost of the non-logic propagation delay for the asynchronous implementation is large. For ideal synchronous logic, the non-logic delay includes only the propagation delay of the memory elements. For asynchronous implementation, it includes both the latch propagation delay and the time it takes to complete the final two cycles of the four-cycle handshake. For this reason, the propagation delay of the Muller-C element as it

transitions through its states is the driving force in the speed performance of the micropipeline approach.

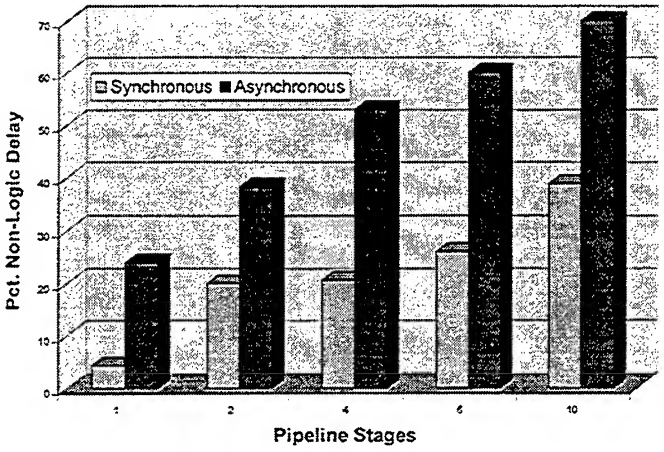

Figure 34. Non-Logic Delay Comparison

B. CHAPTER SUMMARY

The theoretical advantages and disadvantages of asynchronous logic techniques have been demonstrated in this research. A well designed synchronous system will outperform a four-cycle handkshake micropipelined approach when high throughput is the primary system performance requirement. If power consumption is the driving design factor, asynchronous logic techniques are more efficient. In order to provide increased overall system performance for the asynchronous multiplier, the Muller-C element must be redesigned to decrease propagation delays between state transitions. In-depth analysis of relevant design trade-offs

are the basis for professional system engineering. The synchronous versus asynchronous comparison of this research proved to be a fruitful trade-off analysis.

VI. SUMMARY AND CONCLUSIONS

A. RESEARCH SUMMARY AND CONCLUSIONS

The background and relative virtues of synchronous and asynchronous logic design have been thoroughly investigated throughout this research. The synchronous approach is a mature technique that continues to dominate the digital design field. Asynchronous logic design displays merit when the synchronous disadvantages of clock skew, worst-case system performance, power consumption, modularity, and technology migration are considered. However, circuit complexity, size, and throughput performance hinder the ability of asynchronous techniques to carve out a larger portion of the digital design market.

Theoretical timing calculations and Tanner SPICE simulations have been utilized to determine the proper function and configuration of the four-cycle micropipelined asynchronous technique. Analysis and comparison of an 8x8-bit multiplier developed utilizing this local timing method versus one implemented with synchronous logic highlight theoretical assumptions. As expected, the asynchronous technique consumes less power than an optimum synchronous multiplier across five separate pipelined implementations. However, the time lost completing all four cycles of the micropipeline handshake limit the throughput rate of the

asynchronous system. Therefore, the speed performance of the synchronous implementation is superior. In the end, a design engineer has been handed the tools required to make a critical decision; what is more important, system speed performance or power consumption?

B. AREAS FOR FUTURE STUDY

This research has concentrated on the schematic design of a moderately complex integrated circuit. Layout, fabrication, and test of optimum synchronous and asynchronous pipelined multipliers would further enhance the knowledge base for technique comparisons.

As discussed in Chapter II, another locally timed approach, the two-cycle micropipeline handshake, may save some of the throughput period required by the four-cycle method. As the name suggests, the two-cycle approach does not require the last two elements of the handshake scheme discussed in this thesis and therefore should display improved speed characteristics. Two-rail asynchronous design techniques are also available and have been investigated in other areas. Schematic design of either of these two systems is another future research opportunity with the two-cycle handshake showing the most promise for system performance improvement.

Finally, it may be possible to tighten up the four-cycle handshake by utilizing the current-mode latch for both

the memory elements and the Muller-C element. The decreased propagation delay in the memory elements would not afford too much performance improvement; however, a CML clocked latch could be utilized vice a cross-coupled NOR SR latch to improve the propagation delays inherent in the control handshake.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Loomis, H. *Class Notes for EC 4830*, Naval Postgraduate School, Monterey, CA, Spring Quarter, 1999.

2. Poole, N.R., " Self-Timed Logic Circuits" , *Electronics and Communications Engineering Journal*, pp261-270, Dec 1994.

3. Calvert, J.R., *Design of a Synchronous Pipelined Multiplier and Analysis of Clock Skew in High-Speed Digital Systems*, Masters Thesis, Naval Postgraduate School, Monterey, CA, Dec 2000.

4. Buford, R.G., and others, " An 180MHz 16 bit Multiplier Using Asynchronous Logic Design Techniques" , Custom Integrated Circuits Conference, Proceedings of the IEEE, pp215-218, May 1994.

5. Grass, E. and Jones, S., " Asynchronous Circuits Based on Multiple Localised Current-Sensing Completion Detection" , Proceedings, Second Working Conference on Asynchronous Design Methodologies, pp170-177, 30-31 May 1995.

6. Acosta, A.J. and others, " Design and Characterization of a CMOS VLSI Self-timed Multiplier Architecture Based on bit -level Pipelined-array Structure" , IEEE Proceedings-Circuits Devices Systems, Vol. 145, No. 4, pp247-253, Aug 1998.

7. Pang, Y. W. and Choy, C. S., " An Asynchronous Matrix Multiplier" , IEEE Region 10 International Conference on Microelectronics and VLSI, pp315-318, 6-10 Nov. 1995.

8. Hauck, S., " Asycnhronous Design Methodologies: An Overview" , Proceedings of the IEEE, Vol. 83, No. 1, pp69-93, Jan 1995.

9. Pollard, L.H., *Computer Design and Architecture*, Prentice-Hall,Feb 1990.

10. Furber, S.B., " Breaking Step - The Return of Asynchronous Logic" , IEE Colloquium on Design and Test of Asynchronous Systems, pp1-4, 1996.

11. Nanda, K., Desai, S.K., and Roy, S.K., " A New Methodology for the Design of Asynchronous Digital Circuits" , 10th International Conference on VLSI Design, pp342-347, Jan 1997.

12. Sutherland, I. E, " Micropipelines" , *Communications of the ACM*, Vol. 32, No. 6. pp720-738, June 1989.

13. Furber, S.B., and Day, P., " Four-Phase Micropipeline Latch Control Circuits" , IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 4, No. 2, pp247-253, Jun 1996.

14. Weste, N.H., and Eshraghian, K., *Principles of CMOS VLSI Design*, 2nd Edition, Addison-Wesley Publishing Co., 1993.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Ste 0944
   Fort Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5101

3. Director, Training and Education 1
   MCCDC, Code C46
   1019 Elliot Rd.
   Quantico, VA 22134-5027

4. Director, Marine Corps Research Center 2
   MCCDC, Code C40RC
   2040 Broadway Street
   Quantico, VA 22134-5107

5 Marine Corps Tactical System Support Activity 4
   Technical Advisory Branch
   Attn: Librarian
   Box 555171
   Camp Pendleton, CA 92055-5080

6. Marine Corps Representative 1
   Naval Postgraduate School
   Code 037, Bldg. 330, Ingersoll Hall, Room 116
   555 Dyer Road
   Monterey, CA 93943

7. Chairman, Code EC 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

8. Engineering and Technology Curricular Office, Code 34 1
   Naval Postgraduate School
   Monterey, CA 93943-5109

9. Professor Douglas Fouts, Code EC/Fs 1
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943-5121

10. Professor Herschel Loomis, Code EC/Lm 1
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943-5121

11. LtCol Kirk Shawhan (USMC) 1
PO Box 749
Quantico, VA 22134-0749

11. Maj. John Calvert (USMC) 1
1422 Woodway Dr.
Ooltewah, TN. 37363